SRI International

FAST PARALLEL SURFACE INTERPOLATION WITH APPLICATIONS TO DIGITAL CARTOGRAPHY

Technical Note No. 470

June 1989

By: Richard Szeliski

Artificial Intelligence Center
Computer and Information Sciences Division

The research reported herein was supported by DARPA Contract DACA76-85-C-0004 (SRI Project 8388).

"The views, opinions, and findings contained in this paper are those of the author(s) and should not be construed as an official Department of Defense position, policy, or decision, unless so designated by other official documentation."

SRI International®

333 Ravenswood Ave. • Menlo Park, CA 94025
(415) 326-6200 • TWX: 910-373-2046 • Telex: 334-486

Report Documentation Page

Form Approved
OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
JUN 1989		**00-06-1989 to 00-06-1989**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Fast Parallel Surface Interpolation with Applications to Digital Cartography	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **SRI International,333 Ravenswood Avenue,Menlo Park,CA,94025**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**40**	

Fast Parallel Surface Interpolation with Applications to Digital Cartography

Richard Szeliski

Artificial Intelligence Center
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025

June 16, 1989

Abstract

The manipulation of two dimensional elevation maps is an important part of digital cartography. In many situations, these maps are computed by interpolating sparse data such as isolated elevation points obtained from stereo matching. In this paper, we present a surface interpolation algorithm based on variational splines which is well suited to massively parallel computers. Using multiresolution parallel relaxation, we can efficiently compute the interpolated surface and also have local control over its continuity and smoothness. We apply this technique to sparse elevation data and to elevation contours, and show how to add realistic fractal detail through stochastic relaxation. We also present a multiresolution decomposition algorithm and a fast parallel 3-D rendering algorithm.

1 Introduction

Digital cartography is an emerging field which concerns itself with the application of computers to the creation and manipulation of maps [Hanson88]. The data being manipulated can be represented in a variety of formats; of these, the digital terrain model (DTM) is one of the most common and most general [USGS87]. A DTM represents the data using a two-dimensional array which encodes some local feature such as elevation, terrain classification, or cultural status.

The creation of digital terrain models often involves the interpolation of sparse elevation data obtained through field surveys or aerial photogrammetry [Slama80]. Other common operations include the alignment of data taken from different samples or sources, and the automatic correction of errors or biases in the data. A variety of techniques have been developed for doing surface interpolation [Fuchs77, Bartels87, Cendes87]. In this paper, we will introduce a new class of interpolators based on variational splines [Ahlberg67]. These interpolators are extremely flexible and allow local control over quantities such as continuity, orientation, and smoothness. The variational splines are formulated using an optimization framework which specifies the desirable properties for the interpolated solution (e.g., the closeness of fit to the data and/or the smoothness of the solution).

The variational splines that result from this framework can be computed in two different ways. The first involves triangulating the two-dimensional domain of the digital terrain model using the sparse data points as corners and solving for the coefficient of the polynomial (spline) patches on each triangle. The second involves using a fine rectangular discretization which corresponds to the final resolution of the map. This approach allows us to combine information from various sources (e.g., low-resolution elevation maps obtained through photogrammetry with high-precision survey measurements) and to have local control over the model properties. This latter approach is the one which we adopt in this paper.

To convert the analytic (continuous) equations describing the variational spline into a discrete set of equations, we use either finite difference or finite element analysis. The resulting set of equations can then be solved using parallel relaxation. These iterative algorithms solve the system of equations by repeatedly setting each grid point to a weighted average of its neighbors. They are thus extremely well suited to massively parallel architectures such as the Connection Machine^{TM} [Hillis85].

Unfortunately, the convergence of these relaxation algorithms can be quite slow. In this paper, we show how multiresolution methods such as multigrid and preconditioning with hierarchical basis functions can be used to accelerate this convergence. The resulting algorithms have a performance which makes them suitable for real applications.

In addition to developing an efficient surface interpolation algorithm, we examine other parallel algorithms for map manipulation. We show how a stochastic (random) generalization of surface interpolation can be used to add fractal detail to our surface. This same approach can also be used to quantify the uncertainty in our map estimates. We examine how to provide a multiresolution description of the terrain model, which may be useful in the process of extracting a reduced description based on prominent features. We also develop a fast parallel rendering algorithm for displaying three-dimensional views of the surface.

The algorithms described in this paper have all been implemented on the Connection Machine. Their suitability, however, is not limited to just fine-grained (SIMD) architectures [Potter85]. Because of their regular structure, our algorithms are well suited to medium-grained (MIMD) architectures and even vector or array processors. Some of these algorithms have also been implemented on standard RISC architectures with good performance. Devel-

oping massively parallel algorithms thus allows us to use a wide range of current architectures while anticipating the increasing parallelism of future computing hardware.

1.1 Overview

This paper starts in Section 2 with a review of surface interpolation using variational splines. We present a mathematical formulation of the problem and discuss surface interpolation vs. surface approximation. We also show how to achieve local control over both smoothness and continuity. Our variational formulation is compared to a Bayesian (probabilistic) formulation. In Section 3, we derive the discrete equations for the surface (elevation map) and show how they can be solved using local relaxation. We discuss two different acceleration methods: multigrid relaxation, and conjugate gradient descent preconditioned with hierarchical basis functions. In Section 4, we discuss how interpolation can be used for amplification, i.e., to increase the resolution of the data and to fill in unknown areas. We also show how to add fractal detail to the surface to increase the realism and how to use such random samples to obtain a Monte Carlo esimate of the local uncertainty (variance) in the solution. Section 5 examines the opposite problem, that of reducing a terrain map to a more abstract or compact form. We discuss how a relative multiresolution representation can be used to aid this processes. Section 6 describes an efficient parallel algorithm for rendering 3-D views of the terrain model with optional texture or intensity mapping from aerial images. We conclude the paper with a discussion of the advantages and disadvantages of the two-dimensional terrain model representation.

2 Surface interpolation with variational splines

The problem of interpolating a two-dimensional surface such as an elevation map through a set of data points is usually underconstrained, i.e., there are many possible surfaces which pass through the given points. One way of resolving this problem is to heuristically choose a good interpolating algorithm, e.g., to triangulate the domain and use piecewise linear patches. Another possibility is to cast surface interpolation as an optimization problem, e.g., to maximize the smoothness of the surface while minimizing the error of the fit to the data points. This latter approach, which is called variational spline fitting [Ahlberg67], is the one we adopt in this paper.

To formally characterize the optimization problem, we use *regularization* theory [Tikhonov77]. This mathematical technique imposes a weak smoothness constraint on the possible solutions. The functional to be minimized

$$\mathcal{E}(f) = \mathcal{E}_d(f; \{p_i\}) + \lambda \mathcal{E}_s(f) \tag{1}$$

is thus the weighted sum of two terms: the data compatibility constraint $\mathcal{E}_d(f; \{p_i\})$ and the smoothness constraint $\mathcal{E}_s(f)$. The regularization parameter λ is used to adjust the closeness

of the fit between the surface and the data. Regularization has recently been applied to a wide range of problems in low-level computer vision [Poggio85a, Terzopoulos86a].

2.1 Interpolation vs. approximation

The data compatibility constraint measures the distance between the collection of depth values $\{p_i\} = \{(u_i, v_i, d_i)\}$ and the interpolated surface $f(u, v)$. In its simplest form, this constraint is a weighted squares energy measure

$$\mathcal{E}_{\mathrm{d}}(f; \{p_i\}) = \frac{1}{2} \sum_i c_i \left(f(u_i, v_i) - d_i\right)^2 \tag{2}$$

where the weights c_i are inversely related to the variance of the measurements ($c_i = \sigma_i^{-2}$). This weighting of the data allows us to combine both low-resolution data, such as that obtained from large-scale photogrammetry, with high-resolution data obtained from field surveys or more detailed photosurveys. It also allows us to take into account the inherently uncertain nature of elevation measurements.

If we wish the surface to pass exactly through the data points (i.e., to have an *interpolating* instead of an *approximating* solution), we can set the weights c_i to a very large value, or equivalently, we can set the regularization parameter $\lambda \to 0$. This does not pose any significant complications in our framework (it is easy to represent sufficiently small or large values using floating point numbers), but it may slow down the convergence of certain algorithms such as conjugate gradient if we do not use the correct preconditioning (Section 3). In practice, we almost never desire a true interpolated solution since our data points are never without error.[1]

Choosing an optimal value of λ is an active area of research in regularization theory. A powerful method for choosing this parameter is called generalized cross-validation [Craven79]. It involves minimizing the distance between each data point d_i and the spline approximation fit to the remaining data points. This approach is quite expensive, since for each sample value of λ, n spline fits must be calculated, where n is the number of data points. A simpler approach is simply to adjust λ until the variance of the residuals $r_i = f(u_i, v_i) - d_i$ matches that of the noise model (σ_i^2). A related method is process whitening [Maybeck82, Marroquin85], which chooses a λ that makes the residuals as uncorrelated as possible. A number of other approaches to this problem, including a new method based on maximum likelihood estimation, are discussed in [Szeliski88].

[1] We could also formulate the regularization problem as $\min_f \mathcal{E}_{\mathrm{s}}(f)$ such that $|f(u_i, v_i) - d_i| < \epsilon$ for all i [Poggio85a]. However, problems of this sort are more difficult to solve and the utility of this formulation seems questionable.

2.2 Smoothness and continuity

The smoothness constraint (or *stabilizer*, in regularization theory) is a functional or norm of $f(u,v)$ that encodes the *variation* in the surface. Two examples of possible smoothness functionals are the *membrane* model

$$\mathcal{E}_s(f) = \frac{1}{2}\int\int \left(f_u^2 + f_v^2\right)\, du\, dv,$$

which is a small deflection approximation of the surface area, and the *thin plate* model

$$\mathcal{E}_s(f) = \frac{1}{2}\int\int \left(f_{uu}^2 + 2f_{uv}^2 + f_{vv}^2\right)\, du\, dv,$$

which is a small deflection approximation of the surface curvature (the subscripts indicate partial derivatives) [Terzopoulos86b]. These two models can be combined into a single functional

$$\mathcal{E}_s(f) = \frac{1}{2}\int\int \rho(u,v)\{[1-\tau(u,v)][f_u^2 + f_v^2] + \tau(u,v)[f_{uu}^2 + 2f_{uv}^2 + f_{vv}^2]\}\, du\, dv \qquad (3)$$

where $\rho(u,v)$ is a "rigidity" function, and $\tau(u,v)$ is a "tension" function . The rigidity and tension functions can be used to allow depth ($\rho(u,v) = 0$) and orientation ($\tau(u,v) = 0$) discontinuities. The above smoothness constraint corresponds to a "generalized piecewise continuous spline under tension" [Terzopoulos86b].

To see the effects of using various smoothness constraints, consider the nine data points shown in Figure 1a. The interpolated solution using a membrane model is shown in Figure 1b. Note how the surface has visible peaks and dips corresponding to the location of the data points. In practice, the membrane interpolator is not sufficiently smooth for cartographic applications. The interpolated solution using a thin plate model is shown in Figure 1c. This model may sometimes prove to be too smooth, but is often a good choice. The controlled-continuity thin-plate is shown in Figure 1d. Note that a depth discontinuity has been introduced along the left edge and an orientation discontinuity along the right.

2.3 Local control

The stabilizer $\mathcal{E}_s(f)$ described by (3) is an example of the more general controlled-continuity constraint

$$\mathcal{E}_s(f) = \frac{1}{2}\sum_{m=0}^{p}\int w_m(u,v) \sum_{j+k=m} \frac{m!}{j!k!}\left|\frac{\partial^m f(u,v)}{\partial u^j \partial v^k}\right|^2 du\, dv \qquad (4)$$

This formulation allows us to have local control over the continuity and the smoothness of the interpolated solution. The values for the continuity and smoothness can be derived from other cartographic features such as rivers, peaks or cultural features. We will present some graphical examples of this local control in Section 4.

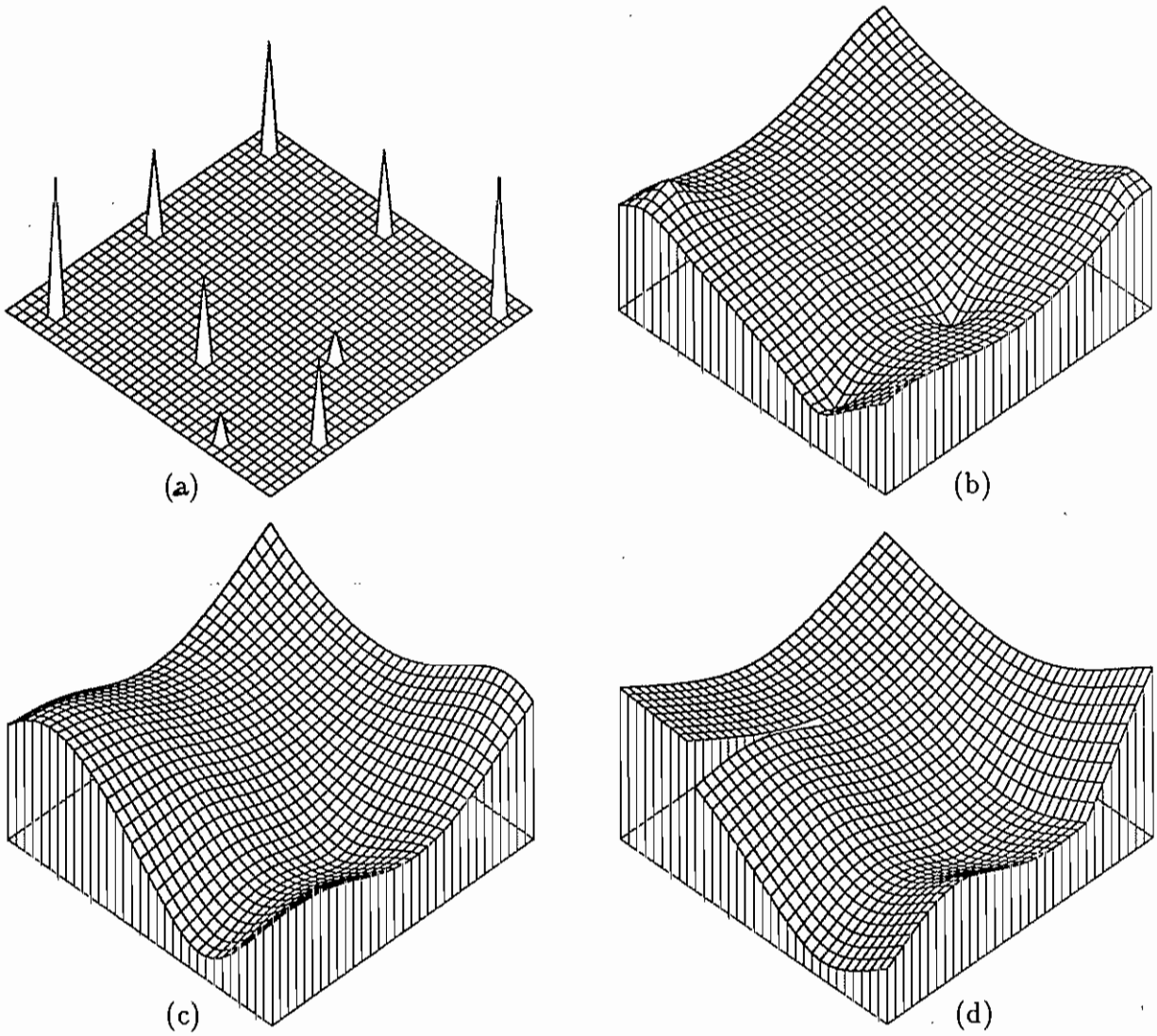


Figure 1: Sample data points and interpolated solutions
(a) sample data points (b) membrane interpolant (c) thin plate interpolant (d) thin plate with discontinuities

Discontinuities in the elevation map (tears) or in its orientation (creases) can be achieved by locally setting $w_1 = w_2 = 0$ or $w_2 = 0$, respectively. These discontinuities usually occur along lines or curves such as cliffs (for tears) or ravines and ridges (for creases).[2] In the discrete formulation of our problem (Appendix A), these features are encoded using binary line and crease variables.

Smooth variations in the w_m correspond to local variation in surface roughness.[3] These can be obtained from a knowledge of the local topography (mountainous, rolling, flat, etc.). We can also derive these parameters through a statistical analysis of our terrain model [Anderssen74, Pentland84, Hewett86]. In the discrete formulation, local smoothness control is achieved by making the w_m's discrete fields coincident with the elevation map (Appendix A). The controlled-continuity constraint (4) can also be extended to model anisotropic surfaces [Boult86], i.e., surfaces such as parallel mountain ranges whose correlation or structure is not rotationally symmetric. In our framework, this could be achieved by using a separate coefficient in front of each partial derivative.

Lastly, we can incorporate non-linear constraints into our regularization framework. For example, we could constrain the elevation map to have a local maximum at a known peak location. We could also constrain certain areas of the map (such as man-made areas or lakes) to be flat. These constraints result in discrete equations that are no longer linear and are not directly amenable to the same optimization techniques which we study in this paper. However, since the algorithms which we develop are iterative, the extension to non-linear constraints is not too difficult [Platt88]. This should be a fruitful direction for future research.

2.4 Relationship to Bayesian estimation

The energy-based framework for surface interpolation can be related to a probabilistic (Bayesian) framework through the use of Gibbs distributions [Geman84]. In the Bayesian framework, surface interpolation is cast as an optimal estimation problem. A *prior model*, $P(f)$, provides a statistical description of the surface which we are trying to reconstruct. A *sensor model*, $P(\{p_i\}|f)$, describes the noisy process by which the data points were sampled from the surface. From these two distribution, we can compute the *posterior model*, $P(f|\{p_i\})$, using Bayes' Rule

$$P(f|\{p_i\}) = \frac{P(\{p_i\}|f)\,P(f)}{P(\{p_i\})}, \tag{5}$$

where $P(\{p_i\})$ is used to normalize the posterior distribution. We can then compute the *Maximum a Posteriori* (MAP) estimate by finding the value of f which maximizes (5).[4]

[2]The problem of automatically detecting discontinuities in surfaces has been extensively studied in the field of computer vision [Marroquin84, Blake87, Terzopoulos86b, Leclerc89].

[3]This is equivalent to having a spatially varying λ.

[4]Other estimates can also be obtained from the posterior distribution [Marroquin85, Szeliski88], but for our application they are equivalent to the MAP estimate.

We can relate the Bayesian framework to the energy-based regularization framework by setting

$$P(f) \propto \exp\left(-\mathcal{E}_s(f)\right) \tag{6}$$

and

$$P(\{p_i\}|f) \propto \exp\left(-\mathcal{E}_d(f;\{p_i\})\right). \tag{7}$$

For this choice of prior and sensor models, we find that

$$P(f|\{p_i\}) \propto \exp\left(-\mathcal{E}(f)\right) \tag{8}$$

and that maximizing the posterior probability $P(f|\{p_i\})$ is equivalent to minimizing the energy $\mathcal{E}(f)$. The MAP estimate therefore corresponds to the usual minimum energy solution obtained from regularization.

This observation has been made before in the regularization literature [Kimeldorf70], but it has not been exploited to a great degree. One way we can make use of this relationship between energies and log probabilities is to build better sensor models. For example, the probability distribution (7) corresponding to our weighted squares data constraint (2) is

$$P(\{p_i\}|f) = \prod_i \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{[d_i - f(u_i, v_i)]^2}{2\sigma_i^2}\right). \tag{9}$$

From this, we see that our data constraint corresponds to an assumption that each elevation d_i is obtained by sampling the surface f at location (u_i, v_i) and adding independent Gaussian noise with variance σ_i^2. More sophisticated sensor models which model gross errors or three-dimensional uncertainty in position can be developed and integrated into the variational framework by setting the data constraint energy to the negative log of the sensor probability distribution [Szeliski88].

The prior model obtained by exponentiating a stabilizer of the form (4) is a correlated Gaussian field whose spectrum is determined by the order of the stabilizer (i.e., the choice of w_m's) [Szeliski87]. For the membrane and thin plate models, the prior model is a stochastic fractal, i.e., a random surface which is self-affine over scale [Mandelbrot82, Voss85]. While this observation may sound counter-intuitive, it provides a statistical model for the surfaces which we are interpolating. Having such a model allows us to generate typical samples from this distribution which we can use to check the validity of our prior model [Geman84].

Under the Bayesian interpretation, the posterior distribution $P(f|\{p_i\})$ describes a family of random surfaces drawn from the prior model which are consistent with the observed data. The most likely sample from this distribution, the MAP estimate, corresponds to the minimum energy solution of (1), as shown in Figure 1d. A *typical* sample from this distribution, which can be generated using stochastic relaxation (Section 3), is shown in Figure 2. We can use such random samples to add realism to our interpolated surfaces and to calculate the uncertainty in our surface estimates (Section 4).

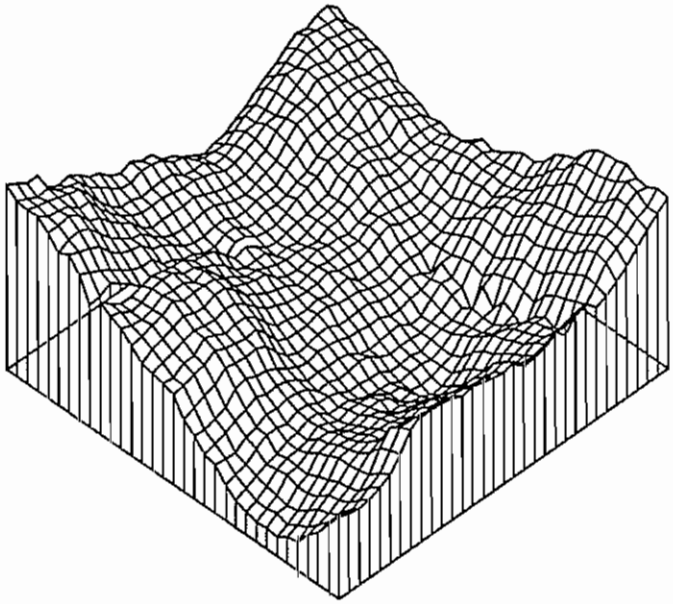


Figure 2: Random sample from posterior distribution

3 Parallel relaxation algorithms

3.1 Finite element discretization

To minimize (1), we apply the finite element method, which provides a systematic approach to the discretization and solution of variational spline problems [Terzopoulos83]. We discretize $f(u,v)$ on a regular fine-grained mesh of *nodal variables* which coincides with the digital terrain model we wish to produce. The advantage of such a representation is that it is independent of the location of the data points. We can thus add more points or combine data from different resolutions without changing the algorithm. It is also easier to represent discontinuities and local variations in smoothness in this form. The regular fine-grained nature of the mesh leads naturally to simple local parallel algorithms which can execute on a massively parallel array of processors.[5]

Using a triangular conforming element for the membrane and a non-conforming rectangular element for the thin plate [Terzopoulos83], we can derive the energy equations

$$E_s(\mathbf{x}) = \frac{1}{2}\sum_{(i,j)}[(x_{i+1,j} - x_{i,j})^2 + (x_{i,j+1} - x_{i,j})^2] \tag{10}$$

[5]The alternatives to our fine-grained discretization include finite elements defined over a data-dependent triangulation of the domain [Fuchs77, Cendes87] and kernel splines [Boult86]. Non-regular (adaptive) grids could also be used to give increased resolution where needed.

for the membrane (the subscripts indicate spatial position) and

$$E_s(\mathbf{x}) = \frac{1}{2}h^{-2}\sum_{(i,j)}[\ (x_{i+1,j} - 2x_{i,j} + x_{i-1,j})^2 + 2(x_{i+1,j+1} - x_{i,j+1} - x_{i+1,j} + x_{i,j})^2 + (x_{i,j+1} - 2x_{i,j} + x_{i,j-1})^2] \tag{11}$$

for the thin plate, where $h = |\Delta u| = |\Delta v|$ is the size of the mesh (isotropic in u and v). These equations hold at the interior of the surface. Near border points or discontinuities some of the energy terms are dropped or replaced by lower continuity terms (see Appendix A). The equation for the data compatibility energy is simply

$$E_d(\mathbf{x}, \mathbf{d}) = \frac{1}{2}\sum_{(i,j)} c_{i,j}(x_{i,j} - d_{i,j})^2 \tag{12}$$

with $c_{i,j} = 0$ at points where there is no input data.

If we concatenate all the nodal variables $\{x_{i,j}\}$ into one vector $\mathbf{x}$, we can write the prior energy model as one quadratic form

$$E_s(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}_s\mathbf{x} \tag{13}$$

where $\mathbf{x}$ is the vector of nodal variables, i.e., $\mathbf{x} = \{f(hi, hj)\}$. The *stiffness* matrix $\mathbf{A}_s$ is extremely sparse, having at most 13 entries per row. Similarly, the data constraint in (2) can be written as

$$E_d(\mathbf{x}, \mathbf{d}) = \frac{1}{2}(\mathbf{x} - \mathbf{d})^T\mathbf{A}_d(\mathbf{x} - \mathbf{d}), \tag{14}$$

where $\mathbf{d}$ is a zero-padded vector of elevation values, and the diagonal matrix $\mathbf{A}_d$ has entries c_i where data points coincide with nodal variables and zeros elsewhere.

Using (13) and (14), we write the combined energy (1) in discrete form as

$$E(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{x}^T\mathbf{b} + k \tag{15}$$

where k is a constant,

$$\mathbf{A} = \lambda\mathbf{A}_s + \mathbf{A}_d, \quad \text{and} \quad \mathbf{b} = \mathbf{A}_d\mathbf{d}.$$

This energy function has a minimum at $\mathbf{x} = \mathbf{x}^*$, the solution to the linear system of algebraic equations

$$\mathbf{A}\mathbf{x} = \mathbf{b}. \tag{16}$$

3.2 Deterministic and stochastic relaxation

In principle, the solution to (16) can be found using either direct or iterative numerical methods. Direct methods are impractical for solving large systems associated with fine

meshes because of excessive storage requirements.[6] Iterative methods, on the other hand, do not require any additional storage and are amenable to parallel implementations. Here, we will study four iterative methods: Gauss-Seidel relaxation, weighted Jacobi relaxation, conjugate gradient descent, and stochastic relaxation.

Gauss-Seidel relaxation is in principle a sequential algorithm where the nodal variables x_i are updated one at a time. The new value for x_i is chosen to minimize the local energy

$$E(x_i) = \frac{1}{2}a_{ii}x_i^2 + \left(\sum_{j \in N_i} a_{ij}x_j - b_i\right) x_i + k_i, \tag{17}$$

where a_{ij} are the entries of $\mathbf{A}$, and N_i expresses the fact that the a_{ij} are nonzero only for certain neighbors of node i (see Appendix A for details). When using Gauss-Seidel relaxation, we choose the new node value which minimizes this local energy

$$x_i^+ = a_{ii}^{-1}\left(b_i - \sum_{j \in N_i} a_{ij}x_j\right) \tag{18}$$

(for pure interpolation, we set $x_i^+ = d_i$ at points coincident with data). We can thus rewrite the local energy as

$$E(x_i) = \frac{1}{2}a_{ii}(x_i - x_i^+)^2 + k_i'. \tag{19}$$

Gauss-Seidel relaxation is well suited to sequential architectures, since it converges faster than the parallel Jabobi algorithm (see below). On a parallel architecture, we can update nodes simultaneously as long as no two nodes are neighbors of each other ($j \notin N_i$). For the membrane interpolator, this can be achieved by using a red-black coloring (updating the red squares on a checkerboard in parallel, then the black squares) [Briggs87]. For the thin plate, we have to use at least 5 colors, and it is often easier to code the algorithm using 9 colors arranged in a 3×3 array. On a machine like the Connection Machine, this can result in an algorithm where each Gauss-Seidel cycle is 9 times slower than a Jacobi cycle.

Weighted Jacobi relaxation is the fully parallel version of Gauss-Seidel. To ensure that the iterative algorithm converges, we take a smaller step than we would in Gauss-Seidel

$$x_i^+ = x_i + \omega a_{ii}^{-1}\left(b_i - \sum_{j \in N_i'} a_{ij}x_j\right) \tag{20}$$

where the neighborhood N_i' now contains the point i. The underrelaxation parameter ω is chosen so that the algorithm convergences. A good choice for ω can be computed by examining the eigenvalues of the iteration matrix [Young71, Briggs87]. The parallel version of (20) can be written using matrix notation,

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \omega\mathbf{D}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}^k), \tag{21}$$

[6] For an $N \times N$ image, we require $O(N^3)$ storage and $O(N^4)$ operations for a direct solution.

where $\mathbf{D}$ is the matrix of diagonal elements from $\mathbf{A}$ (the superscripts indicate the iteration number). The term $\mathbf{r}^k = \mathbf{b} - \mathbf{A}\mathbf{x}^k$ is called the *residual* and measures the current difference between the right and left hand sides of (16).

While both of these algorithms do a good job when the data is reasonably dense, their performance degrades severely as the distance between data points increases (see [Szeliski89a] for examples). A faster algorithm is conjugate gradient descent [Press86]. In this algorithm, we choose a descent direction $\mathbf{p}$ and take an optimally sized step in this direction

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{p}^k. \tag{22}$$

The step size α^k is chosen to minimize the global energy (15), and can be shown to be

$$\alpha^k = \mathbf{p}^{k^T}\mathbf{r}^k / \mathbf{p}^{k^T}\mathbf{A}\mathbf{p}^k.$$

Initially, the direction $\mathbf{p}$ is taken to be the residual $\mathbf{r}$, which is also the negative gradient of the energy (15). To increase the convergence of the algorithm, we require that successive directions be *conjugate*, i.e., that $\mathbf{p}^{k^T}\mathbf{A}\mathbf{p}^{k-1} = 0$. We can ensure this by setting

$$\mathbf{p}^k = \mathbf{r}^k - \beta^k \mathbf{p}^{k-1} \tag{23}$$

with

$$\beta^k = \mathbf{r}^{k^T}\mathbf{A}\mathbf{p}^{k-1} / \mathbf{p}^{k-1^T}\mathbf{A}\mathbf{p}^{k-1}.$$

As we can see from these operations, this algorithm is well suited to massively parallel architectures such as the Connection Machine which can both compute weighted local averages (e.g., $\mathbf{A}\mathbf{x}$) and global summations (e.g., $\mathbf{p}^T\mathbf{r}$). A full description of the conjugate gradient algorithm is given Table 1 near the end of this section.

The three algorithms which we have just described can be used to find the surface which minimizes the energy (15). In the Bayesian interpretation, this corresponds to the MAP estimate for the surface. If we wish instead to generate a typical sample from the posterior distribution, we can use the Gibbs Sampler [Geman84]. This algorithm is a stochastic generalization of Gauss-Seidel relaxation. Updating the points one at a time, we choose the new value for x_i from the local Boltzmann distribution corresponding to (19)

$$P(x_i) \propto \exp\left(-E(x_i)/T\right) \propto \exp\left(-\frac{1}{2}\frac{(x_i - x_i^+)^2}{T/a_{ii}}\right). \tag{24}$$

This distribution is a Gaussian with mean equal to the deterministic update value x_i^+ and a variance equal to T/a_{ii}. Thus, the Gibbs Sampler is equivalent to the usual Gauss-Seidel relaxation algorithm with the addition of some locally controlled Gaussian noise at each step.[7] The amount of roughness can be controlled with the "temperature" parameter T.

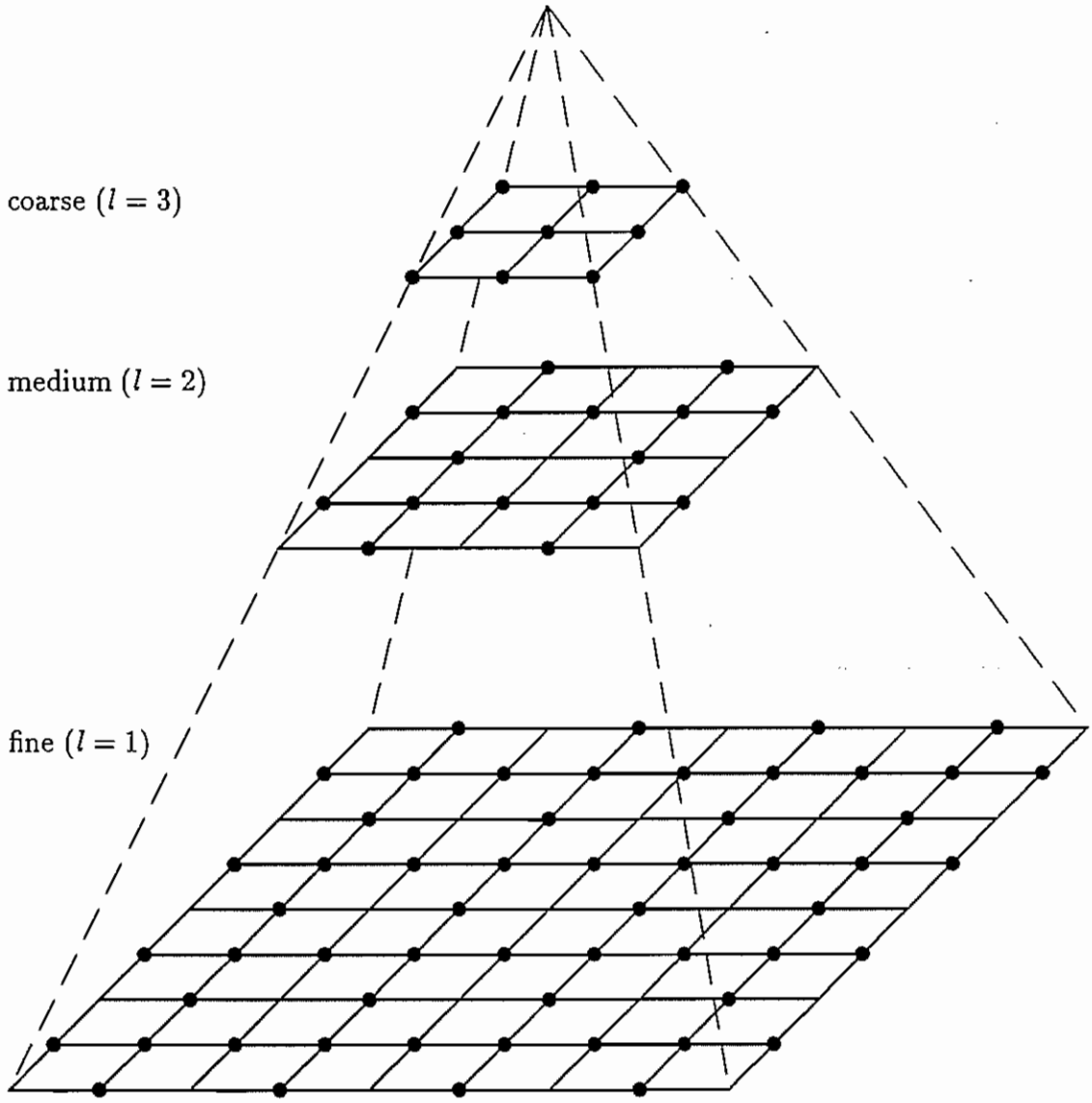


Figure 3: Multiresolution pyramid
The solid dots indicate the nodes in the hierarchical basis.

3.3 Multigrid relaxation

Although the above iterative algorithms will eventually compute the desired solution, the convergence may be unacceptably slow in practice (see [Szeliski89a] for examples). To accelerate convergence, we can use coarse-to-fine relaxation on a multiresolution pyramid (Figure 3). The problem is first solved on a coarser (smaller) mesh, and the solution is used as an initial condition for the next finer level.

This simple idea is often adequate to get a rough solution. However, to achieve maximum accuracy, a more sophisticated approach called *multigrid* is usually required. Multigrid relaxation algorithms [Hackbusch85, Briggs87] are based on the observation that local iterative methods are good at reducing the high frequency components of the interpolation error, but are poor at reducing the low frequency components. By solving a related problem on the coarser grid, this low frequency error can be reduced more quickly.

To develop a multigrid algorithm, several components must be specified: the method used to derive the energy equations at the coarser levels from the fine level equations; a *restriction* operation that maps a solution at a fine level to a coarser grid; a *prolongation* operation that maps from the coarse to the fine level; and a *coordination scheme* that specifies the number of iterations at each level and the sequence of prolongations and restrictions.

A sophisticated version of multigrid relaxation was developed by [Terzopoulos83]. In his implementation, the coarser meshes are coincident with the original mesh, i.e., the coarser level nodes are a subset of the finer level nodes. Simple injection (subsampling) is used for restriction, and third-degree Lagrange interpolation is used for prolongation. A Full Multigrid (FM) method is used to coordinate the inter-level interactions. This ensures that the coarse levels have the same accuracy as the fine level solutions.

Our own experiments with multigrid on the Connection Machine were somewhat disappointing. The convergence rates were not significantly better than those obtained with simple coarse-to-fine relaxation. This problem may be due to the irregular nature of the data constraints (multigrid techniques were originally developed for the solution of uniform smooth partial differential equations). It is quite possible that the use of different (locally adjusted) interpolators could improve the convergence, but we have not investigated this possibility. Instead, we have been developing a multiresolution version of conjugate gradient descent, which we describe next.

3.4 Preconditioned conjugate gradient descent

The conjugate gradient descent algorithm that we described earlier can be accelerated dramatically using the hierarchical basis functions developed recently by [Yserentant86]. In this approach, the usual nodal basis set $\mathbf{x}$ is replaced by a hierarchical basis set $\mathbf{y}$. Certain

[7] A parallel version of stochastic relaxation based on global diffusion is also possible but does not converge as quickly to the equilibrium distribution [Szeliski89b].

elements of the hierarchical basis set have larger support than the nodal basis elements, and this allows the relaxation algorithm to converge more quickly when using the hierarchical set.

To convert from the hierarchical to the nodal basis set, we use a simple linear (matrix) transform

$$\mathbf{x} = \mathbf{S}\mathbf{y}, \tag{25}$$

where $\mathbf{S} = \mathbf{S}_1\mathbf{S}_2 \ldots \mathbf{S}_{L-1}$, and L is the number of levels in the hierarchical basis set. Each of the sparse matrices $\mathbf{S}_l$ interpolates the nodes at level $l+1$ to level l and adds in the nodes corresponding to the new level. The columns of $\mathbf{S}$ give the values of the hierarchical basis functions at the nodal variable locations.

In his paper, Yserentant uses recursive subdivision of triangles to obtain the nodal basis set. The corresponding hierarchical basis then consists of the top-level (coarse) triangularization, along with the subtriangles that are generated each time a larger triangle is subdivided. Linear interpolation is used on a triangle each time it is subdivided. We can generalize this notion to arbitrary interpolants defined over a rectangular grid [Szeliski89a]. Each node in the hierarchical basis is assigned to the level in the multiresolution pyramid where it first appears (the solid dots in Figure 3). This is similar to the multiresolution pyramid, except that each level is only partially populated, and the total number of nodes is the same in both the nodal and hierarchical basis sets. To fully define the hierarchical basis set, we select an interpolation function that defines how each level is interpolated to the next finer level before the new node values are added in.

The resulting algorithms for mapping between the hierarchical and nodal basis sets are simple and efficient. We use

```
procedure S
    for l = L − 1 down to 1
        for i ∈ M_l
            for j ∈ N_i
                u(i) = u(i) + w(i; j)u(j)
end S
```

to convert from the hierarchical to the nodal basis set. In this procedure, which goes from the coarsest level ($l = L$) to the finest ($l = 1$), each node is assigned to one of the level collections $\mathcal{M}_l$. Each node also has a number of neighboring nodes $\mathcal{N}_i$ on the next coarser level that contribute to its value during the interpolation process. The $w(i;j)$ are the weighting functions that depend on the particular choice of interpolation function (these are the off-diagonal terms in the $\mathbf{S}_l$ matrices).

We also use the adjoint of this operation

```
procedure S^T
```

```
for l = 1 to L − 1
    for i ∈ M_l
        for j ∈ N_i
            u(j) = u(j) + w(i; j)u(i)
end S^T
```

in our modified conjugate gradient descent algorithm.

These mapping algorithms are easy to code (once the interpolation functions have been precomputed) and require very few computations to perform. On a serial machine, these procedures use $O(n)$ operations (multiplications and additions), where n is the number of nodes. On a parallel machine, $O(L)$ parallel steps are required, where $L \approx \frac{1}{2}\log n$ is the number of levels in the pyramid. This is the same number of steps as is needed to perform the global summations (inner products) used in the conjugate gradient algorithm. Note that although we have defined the hierarchical basis over a pyramid, it can actually be represented in the same space as the usual nodal basis, and the transformations between bases can be accomplished in place.

The hierarchical basis set allows us to minimize exactly the same energy as the one we derived from the discretization on the finest grid. Substituting $\mathbf{x} = \mathbf{S}\mathbf{y}$ into (15), we obtain the new energy equation

$$\begin{aligned} E(\mathbf{y}) &= \frac{1}{2}\mathbf{y}^T(\mathbf{S}^T\mathbf{A}\mathbf{S})\mathbf{y} - \mathbf{y}^T(\mathbf{S}^T\mathbf{b}) + c \\ &= \frac{1}{2}\mathbf{y}^T\hat{\mathbf{A}}\mathbf{y} - \mathbf{y}^T\hat{\mathbf{b}} + c \end{aligned} \tag{26}$$

where the $\hat{}$ identifies the hierarchical basis vectors and matrices. The advantage of minimizing this new equation is that the condition number of the matrix $\hat{\mathbf{A}}$ is much smaller than that of the original matrix $\mathbf{A}$ [Yserentant86], implying much faster convergence for iterative algorithms such as conjugate gradient.

Unfortunately, the $\hat{\mathbf{A}}$ matrix is not as sparse as the original matrix $\mathbf{A}$, so that a direct minimization of (26) is impractical. Instead, we use the recursive mappings $\mathbf{S}$ and $\mathbf{S}^T$ in conjunction with the original matrix $\mathbf{A}$ and vector $\mathbf{b}$ to compute the required residuals and inner products. The resulting conjugate gradient descent algorithm is identical to the usual single-level algorithm except that we use a smoothed version of the residual $\tilde{\mathbf{r}} = \mathbf{S}\mathbf{S}^T\mathbf{r}$ to choose the new direction [Szeliski89a]. The full algorithm is shown in Table 1.

The above algorithm is an example of the more general idea of preconditioning. One common application of this idea is to divide the residual $\mathbf{r}$ by the diagonal of the $\mathbf{A}$ matrix, i.e., to use $\tilde{\mathbf{r}} = \mathbf{D}^{-1}\mathbf{r}$. This is equivalent to dividing each row and column of the $\mathbf{A}$ matrix by the square root of the diagonal element ($\tilde{\mathbf{A}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$). The new stiffness matrix $\tilde{\mathbf{A}}$ has a lower condition number, especially when the data coupling terms are very stiff (large c_i or small λ). The resulting conjugate gradient algorithm looks a lot like weighted Jacobi with the added conjugation in the descent direction. We can combine the hierarchical basis

0.	initialize r_0 and $\tilde{\mathrm{r}}_0$ using 6. – 8. and set $\mathrm{p}_0 = \tilde{\mathrm{r}}_0$			
1.	w_k	$=$	$\mathrm{A}\mathrm{p}_k$	
2.	α_k^D	$=$	$\mathrm{p}_k \cdot \mathrm{w}_k = \mathrm{p}_k^T \mathrm{A}\mathrm{p}_k$	
3.	α_k^N	$=$	$\mathrm{p}_k \cdot \mathrm{r}_k = \mathrm{p}_k^T \mathrm{r}_k$	
4.†	α_k	$=$	α_k^N / α_k^D	
5.	x_{k+1}	$=$	$\mathrm{x}_k + \alpha_k \mathrm{p}_k$	
6.	r_{k+1}	$=$	$\mathrm{b} - \mathrm{A}\mathrm{x}_{k+1}$	
7.	$\hat{\mathrm{r}}_{k+1}$	$=$	$\mathrm{S}^T \mathrm{r}_{k+1}$	(preconditioning step)
8.	$\tilde{\mathrm{r}}_{k+1}$	$=$	$\mathrm{S}\hat{\mathrm{r}}_{k+1}$	(preconditioning step)
9.	β_{k+1}^N	$=$	$\tilde{\mathrm{r}}_{k+1} \cdot \mathrm{w}_k$	
10.‡	β_{k+1}	$=$	$\beta_{k+1}^N / \alpha_k^D$	
11.	p_{k+1}	$=$	$\tilde{\mathrm{r}}_{k+1} - \beta_{k+1} \mathrm{p}_k$	
12.	loop to 1.			

† $\Delta E(\mathrm{x} + \alpha \mathrm{p}) = \frac{\alpha^2}{2} \mathrm{p}^T \mathrm{A}\mathrm{p} - \alpha \mathrm{p}^T \mathrm{r}$

‡ $(\tilde{\mathrm{r}} - \beta \mathrm{p})^T \mathrm{A}\mathrm{p} = 0$

Table 1: Conjugate gradient descent algorithm

preconditioner with the inverse diagonal preconditioner (using either $\mathrm{D}^{-1/2}\mathrm{S}$ or $\mathrm{S}\mathrm{D}^{-1/2}$) to obtain an algorithm which performs well even when the data coupling is stiff.

3.5 Comparison of algorithms

To compare the performance of the various deterministic relaxation algorithm, we ran these algorithms on the data set shown in Figure 5. First, we computed the optimal solution x^* using 500 iterations of the hierarchical basis conjugate gradient algorithm ($L = 4$). Then, for each of the algorithms tested, we computed the root mean squared (RMS) error

$$e_k = |\mathrm{x}_k - \mathrm{x}^*| / \sqrt{n}$$

as a function of the number of iterations k.

Figure 4 shows the relative performance of the relaxation algorithms, as well as the effects of varying the number of smoothing levels in hierarchical basis conjugate gradient. The topmost curve corresponds to weighted Jacobi relaxation ($\omega = 0.25$), which is the slowest of the algorithms tested. The second curve is for Gauss-Seidel, which initially converges faster but then also tails off. Conjugate gradient does better and continues to converge towards the solution at a reasonable rate. Using preconditioning with hierarchical basis functions dramatically speeds up the convergence of conjugate gradient. Initially, the speedup is proportional to the number of smoothing levels. As the iterations progress, however, the

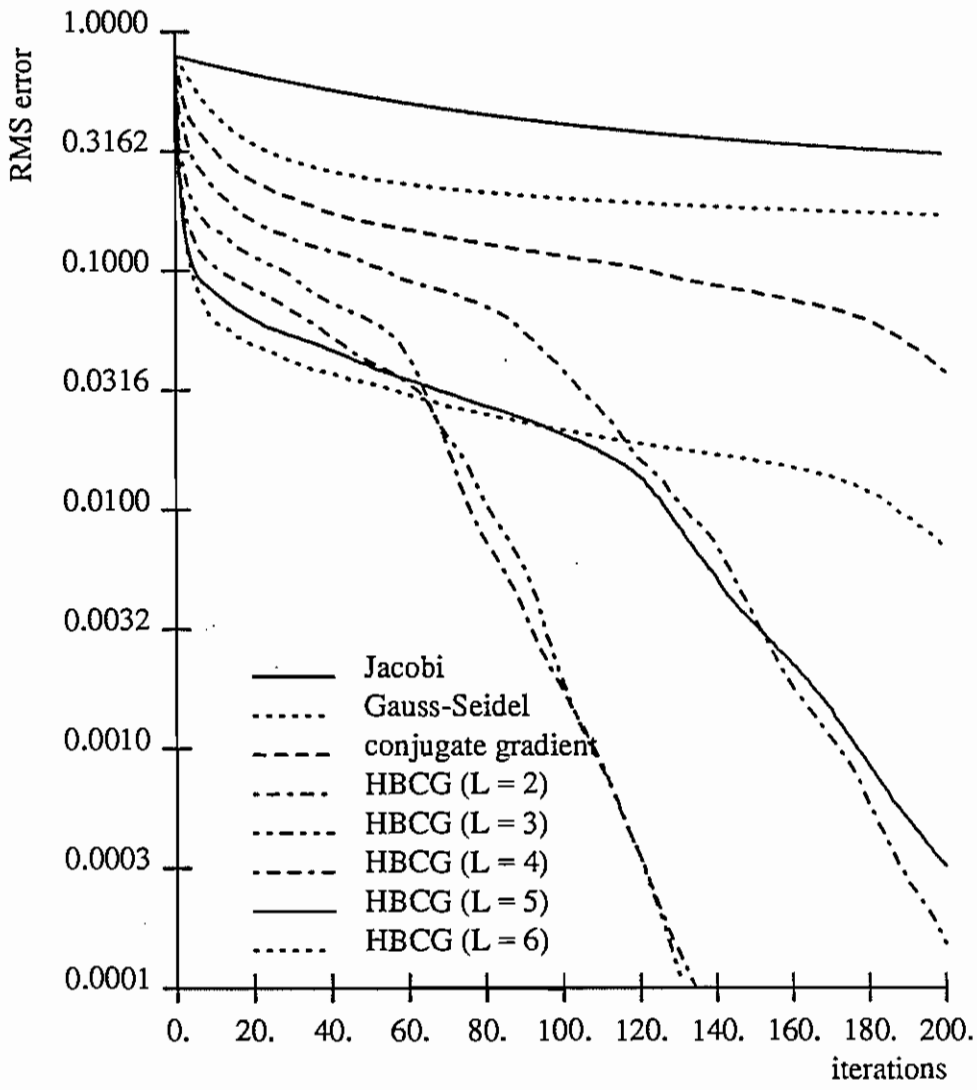


Figure 4: Convergence plot for surface interpolation
The top three curves are for Jacobi, Gauss-Seidel, and conjugate gradient. The lower five curves are for hierarchical basis conjugate gradient with different numbers of smoothing levels L (note that HBCG with $L = 1$ corresponds to conjugate gradient). The fastest convergence is for HBCG with $L = 3$ or $L = 4$.

versions using an intermediate number of smoothing levels ($L = 3$ and $L = 4$) have the best performance. We believe that the optimum number of levels to be used is related to the density of the underlying data points, and we are currently investigating methods for automatically adjusting the smoothing based on the local structure of the data.

4 Data amplification

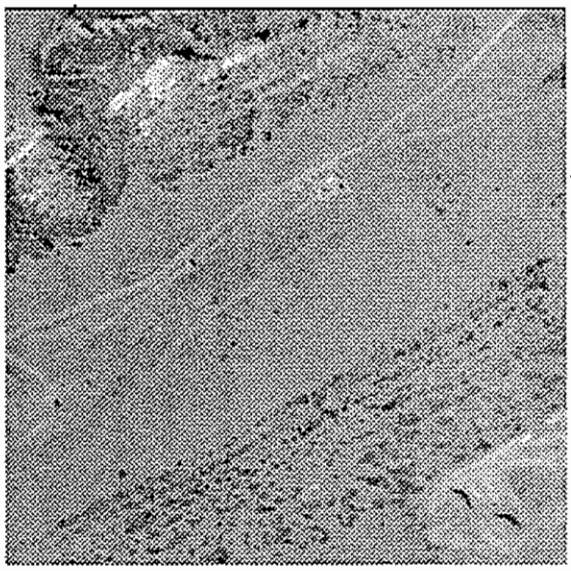
The deterministic relaxation algorithms which we have developed can be used to interpolate smooth terrain models from sparse irregular data. Our stochastic relaxation algorithm can be used to generate random terrain models which fit the data. We can use both of these techniques for *data amplification*, i.e., to produce a denser (and more voluminous) description of our terrain from a sparse set of control points.

4.1 Interpolation from sparse data

The variational splines which we use for surface interpolation are well suited to interpolating sparse irregular elevation data. An example of such data is that obtained from matching isolated features in a stereo pair of aerial images. Figure 5a shows one of the two intensity images which was input to the STEREOSYS stereo matching system developed at SRI [Hannah88]. Figure 5b shows the set of 7696 matches which were produced by this algorithm in a sub-region of this image. These points were then given as sparse elevation constraints to our thin plate interpolating program. The resulting dense elevation map is shown as a contour map in Figure 5c.

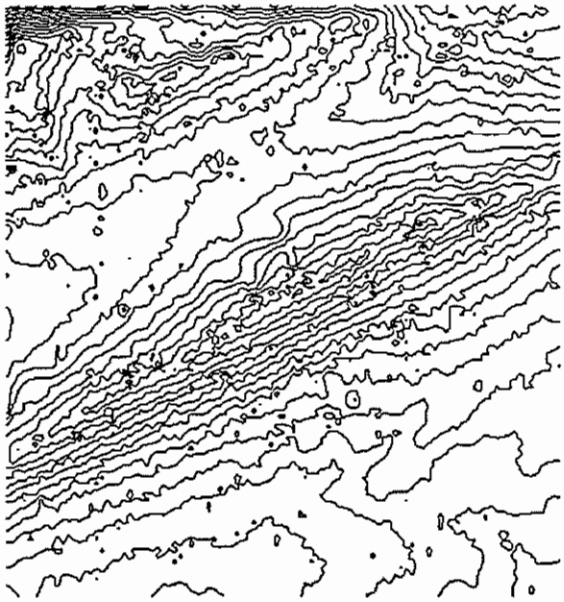
Variational splines can also be used to generate a full digital elevation model from a set of contour lines. Figure 6a shows the original elevation map (from a 1:250,000 scale map of the Rockies just west of Denver) digitized on a 256×256 grid. In this figure, brighter intensities represent higher elevations, and the contour lines are drawn at 100m intervals. This map was subsampled using 200m contour lines, and the resulting sparse data was input to the surface fitting algorithm. The resulting elevation maps are shown in Figures 6b (membrane), 6c (thin plate), and 6d (mixed membrane / thin plate model). Although it is difficult to see from the contour maps, the membrane interpolator ($\{w_0, w_1, w_2\} = \{0, 1, 0\}$) is not smooth enough (locally, it acts like a "soap film"), while the thin plate interpolator ($\{w_0, w_1, w_2\} = \{0, 0, 1\}$) is too smooth (it creates a large "dip" in the middle of the map). Using a blend of these two models ($\{w_0, w_1, w_2\} = \{0, 0.0063, 1\}$) results in a better reconstruction.

We can also use additional (non-topographic) information to control and enhance the surface interpolation process. An example of such information is the hydrography (streams, rivers, and other drainage) that is often available in conjunction with digital elevation models [USGS86]. Figure 7a shows a 256×256 section of a digital elevation model of the Big Basin area; Figure 7b shows the hydrography for the same region. If we interpolate a surface



(a)

(b)



(c)

Figure 5: Surface interpolated from sparse elevation data
(a) sample aerial photograph from stereo pair (b) matched points from stereo pair (c) interpolated thin plate solution on 106×99 grid

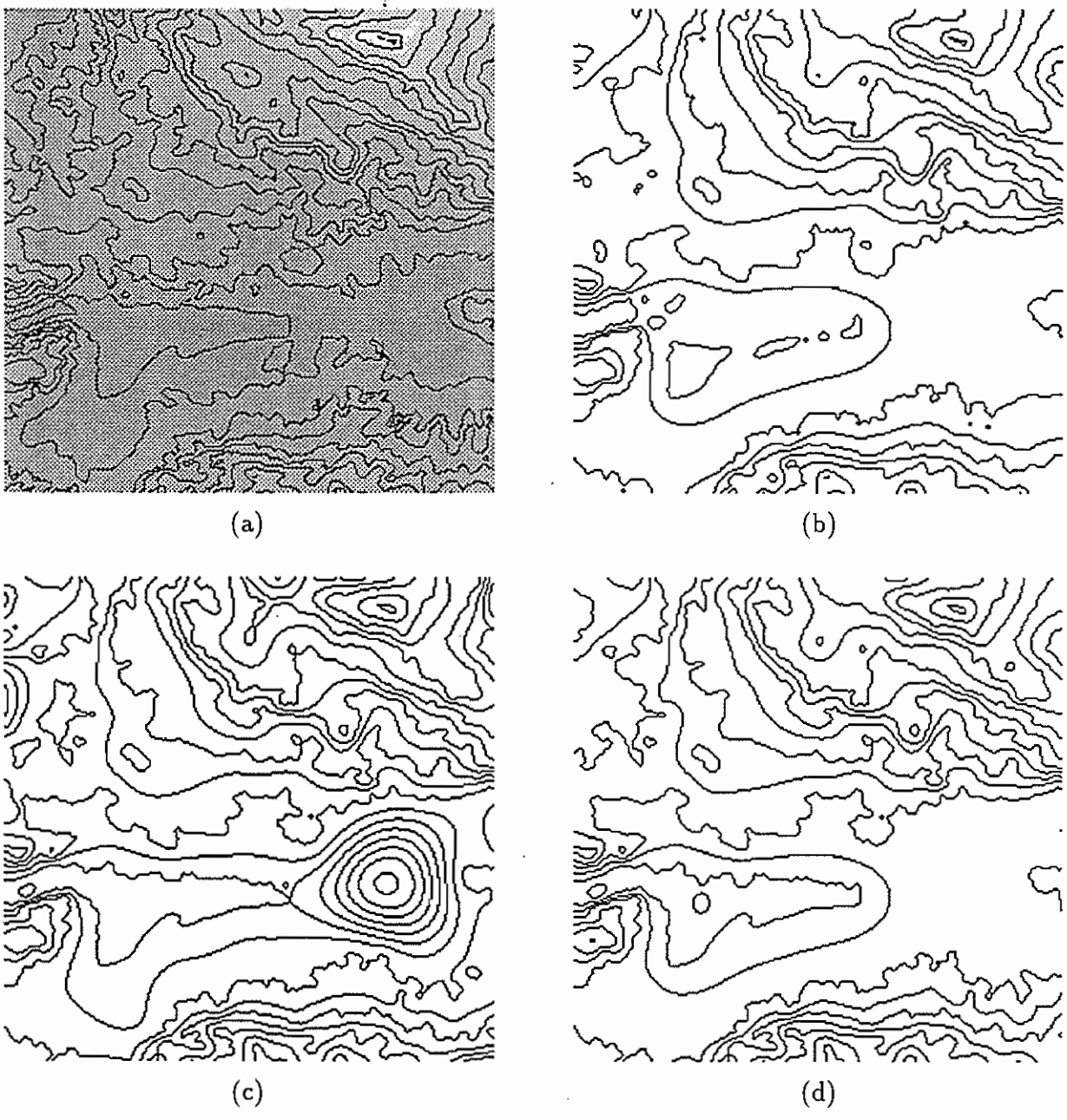


Figure 6: Surface interpolated from contour data
(a) original Colorado Rockies elevation map (100m intervals) (b) membrane interpolated from 200m intervals (c) thin plate interpolated from 200m intervals (d) mixed model interpolated from 200m intervals

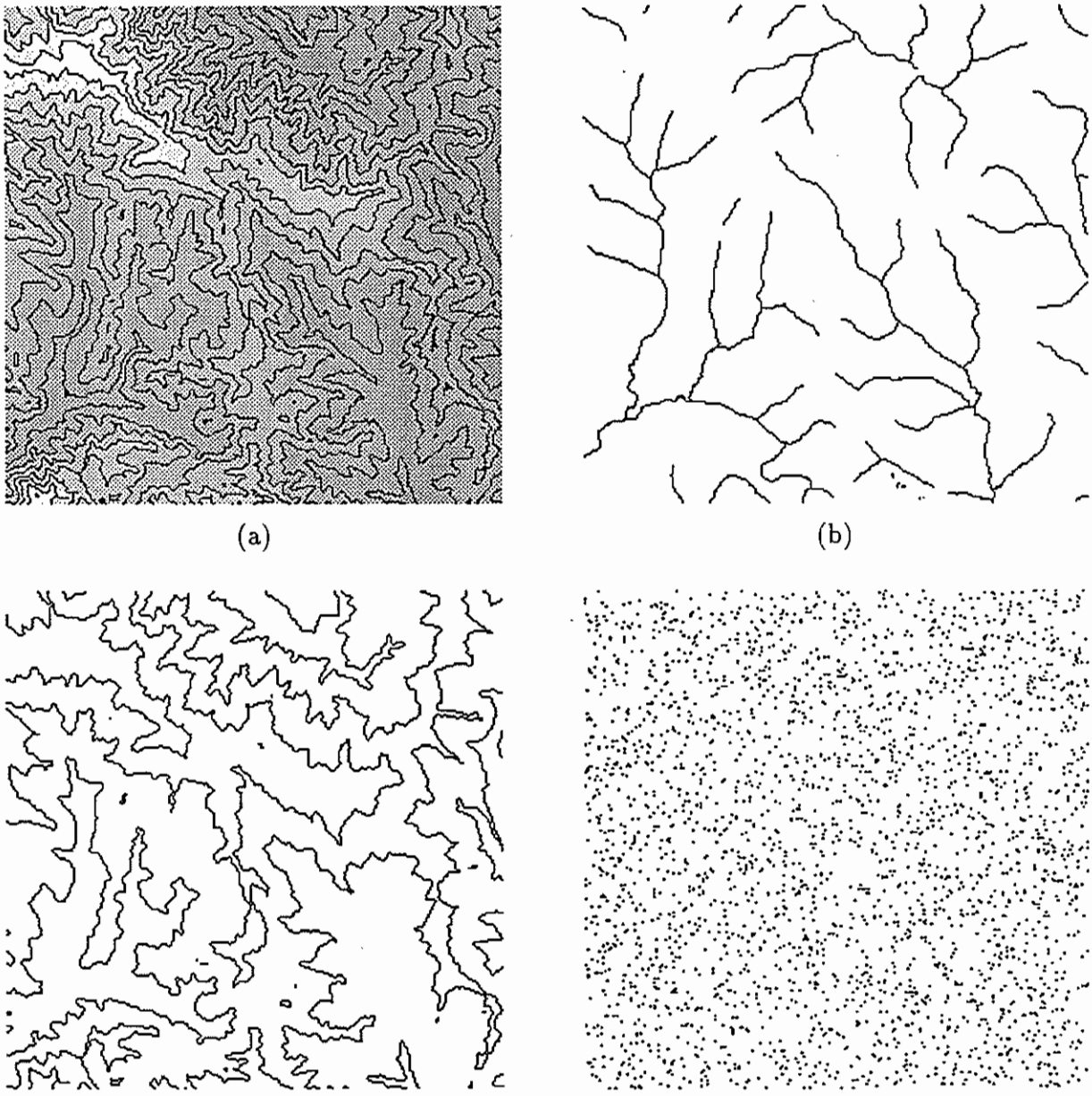


Figure 7: Big Basin elevation map and hydrography
(a) original elevation map (200m intervals) (b) hydrography (drainage) (c) 400m contour lines used for interpolation (d) sparse (2%) sampling of elevation data

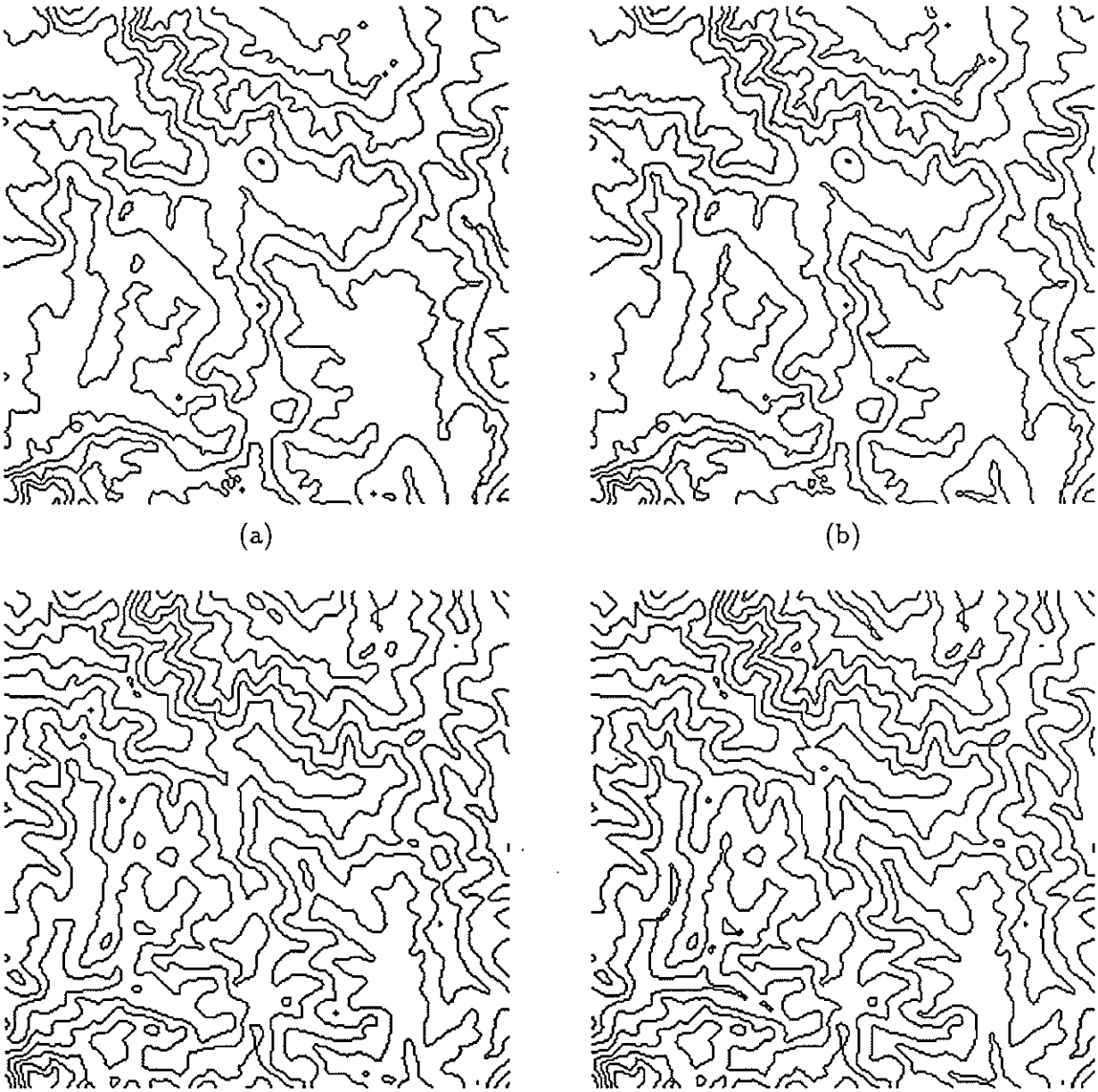


Figure 8: Big Basin reconstructed surfaces
(a) mixed model interpolated from 400m intervals (b) mixed model interpolated with creases from 400m intervals (c) mixed model interpolated from sparse samples (d) mixed model interpolated with creases from sparse samples

directly from the contours shown in Figure 7c, we obtain the surface shown in Figure 8a. If, on the other hand, we use the hydrography to introduce creases (discontinuities in the first derivative), we obtain the more realistic (and accurate) map shown in Figure 8b. A similar effect is shown in Figures 8c and 8d, which were interpolated from the 2% sampling shown in Figure 7d.

For interpolating regularly spaced data or re-sampling grids for rectification, other techniques may be preferable to variational splines. Simple bilinear interpolation or cubic interpolation [Cendes87, Foley87] may suffice. In any event, the interpolation of regularly spaced data with no discontinuities or local variation can always be expressed as a linear shift-invariant filtering operation [Oppenheim75, Poggio85b]. This results in a simpler and quicker algorithm than the one we use in this paper. Variational splines are preferred, however, when there are significant gaps in the data and we desire local control over the continuity.

4.2 Adding fractal detail

Stochastic fractals have become a popular technique in computer graphics for adding realistic detail to rendered images of natural terrain [Fournier82, Mandelbrot82]. Such realism is useful in flight simulators, synthetic scenes used in advertising and movies, and as backgrounds for architectural studies. As we saw in Section 2.4, a random sample from the family of surfaces defined by the variational spline energy has fractal statistics [Szeliski87]. In Section 3, we saw how a simple stochastic extension of relaxation (i.e., the addition of controlled Gaussian noise) can be used to generate fractal surfaces which also match our desired data constraints.

If we apply our stochastic relaxation algorithm to the data shown in Figure 6a, we obtain the contour map shown in Figure 9. Note how the new contour lines look more realistic than the smooth ones shown in Figure 6d. A shaded rendering of this same surface in shown in Figure 10. We can also use our stochastic relaxation with additional inter-level consistency constraints to generate a "zoom sequence" which reveals more detail as the viewer approaches the surface [Szeliski89b].

4.3 Illusion vs. reality

The surface shown in Figures 9 and 10 is, of course, just one possible random sample from a whole family of surfaces which fit the data. The detail in the surface is actually "hallucinated" and does not necessarily correspond to the true detailed shape of the terrain. For certain applications (fly-throughs, mission planning and simulation), having this fictitious detail may be advantageous. For others, however, this detail gives a false sense of confidence in the terrain shape which is not justified by the available data.

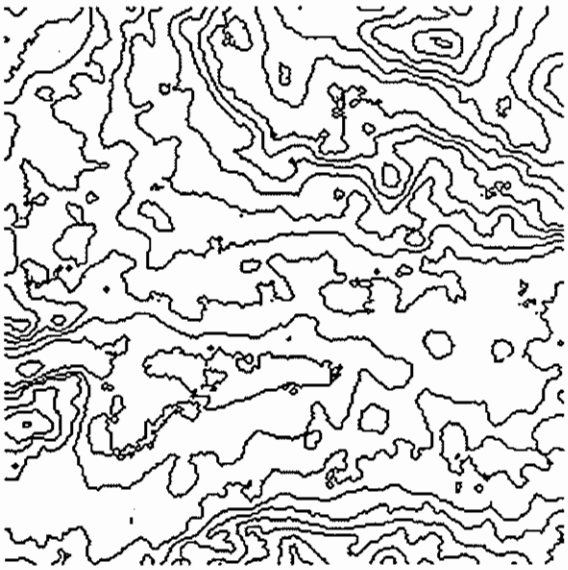


Figure 9: Fractal surface interpolated from contour data

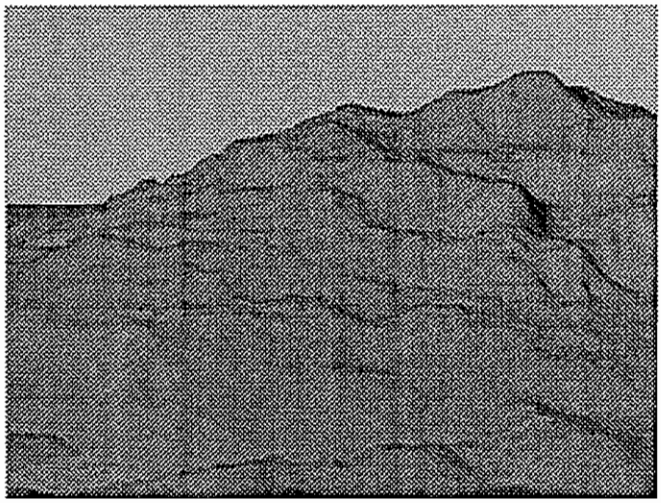


Figure 10: Fractal surface (shaded rendering)

From a Bayesian perspective, the fractal surface of Figure 9 is a *typical* sample from the posterior distribution, whereas Figure 6d is the *most likely* sample (and also the *mean* value). By generating a number of random samples, we can actually compute the local variance in the posterior estimate at each point.[8] These confidence measures can help decide where more measurements are needed or how much to trust the terrain model at a given point.

Displaying the variance as a confidence interval around each surface point is difficult. In one dimension, we can draw an envelope around the curve showing the confidence interval. In two dimensions, we could use color coding to indicate the confidence. If our computation and rendering hardware were fast enough, we could actually display many random samples in succession. The resulting surface would appear to "wiggle" or "shimmer" in areas where it is most uncertain. Whether this additional confidence information will be useful for cartographic applications remains to be seen.

5 Data reduction

The converse of the data amplification process discussed in the previous section is *data reduction*. Here, the aim is to go from a complete description such as an elevation map or terrain model to a more parsimonious description. For example, we may wish to describe the topography by the location of simple features such as peaks, ridges, and valleys. Such sketch maps can be used in the recognition of features and observer location, the semantic interpretation of the terrain model, and the quick generation of rendered views.

The general problem of deriving a reduced description is a difficult one which involves elements of both signal processing and knowledge-based artificial intelligence [Fischler89]. In this section, we will sketch out a more modest proposal which aims to decompose the terrain model into a multiresolution description. This work, while incomplete, may lead to a novel technique for decomposing a surface into a hierarchical description.

5.1 Relative depth representation

The multigrid and coarse-to-fine algorithms described in Section 3.3 keep a full copy of the current depth estimate at each level. An alternative to this absolute multiresolution representation is a *relative* representation [Szeliski88]. In this representation, each level $\mathbf{y}_l$ encodes details pertinent to its own scale, and the *sum* of all the interpolated levels provides the overall depth map estimate

$$\mathbf{x} = \sum_{l=1}^{L} \mathbf{I}_l \mathbf{y}_l \tag{27}$$

where $\mathbf{I}_l$ is the interpolation function associated with each level. The relative representation is thus analogous to a band-pass image pyramid [Burt83, Crowley82], while the absolute

[8] This is a Monte Carlo approach to computing a distribution.

representation is similar to a low-pass pyramid. In the relative representation, each level in the pyramid has its own associated smoothness energy

$$E_s^l(\mathbf{y}^l) = \frac{1}{2}\mathbf{y}_l{}^T \mathbf{A}_s^l \mathbf{y}_l. \tag{28}$$

The data compatibility is measured using the summed depth map $\mathbf{x}$ and the data points $\mathbf{d}$

$$E_d(\mathbf{x}, \mathbf{d}) = \frac{1}{2}(\mathbf{x} - \mathbf{d})^T \mathbf{A}_d(\mathbf{x} - \mathbf{d}). \tag{29}$$

Using $\tilde{\mathbf{y}} = [\mathbf{y}^{1T} \ldots \mathbf{y}^{LT}]^T$ to denote the concatenation of the individual state vectors, $\tilde{\mathbf{I}} = [\mathbf{I}^1 \ldots \mathbf{I}^L]$ for the concatenated interpolation matrices, and

$$\tilde{\mathbf{A}}_s = \begin{bmatrix} \mathbf{A}_s^1 & 0 & \cdots & 0 \\ 0 & \mathbf{A}_s^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{A}_s^L \end{bmatrix} \tag{30}$$

to denote the composite spline model matrix, we can write the overall energy function as

$$\begin{aligned} E(\tilde{\mathbf{y}}) &= \sum_{l=1}^{L} E_s^l(\mathbf{y}^l) + E_d(\mathbf{x}, \mathbf{d}) \\ &= \frac{1}{2}\tilde{\mathbf{y}}^T \tilde{\mathbf{A}}_s \tilde{\mathbf{y}} + \frac{1}{2}(\tilde{\mathbf{I}}\tilde{\mathbf{y}} - \mathbf{d})^T \mathbf{A}_d(\tilde{\mathbf{I}}\tilde{\mathbf{y}} - \mathbf{d}) \\ &= \frac{1}{2}\tilde{\mathbf{y}}^T \tilde{\mathbf{A}} \tilde{\mathbf{y}} - \tilde{\mathbf{y}}^T \tilde{\mathbf{b}} + c \end{aligned} \tag{31}$$

where

$$\tilde{\mathbf{A}} = \tilde{\mathbf{A}}_s + \tilde{\mathbf{I}}^T \mathbf{A}_d \tilde{\mathbf{I}} \quad \text{and} \quad \tilde{\mathbf{b}} = \tilde{\mathbf{I}}^T \mathbf{A}_d \mathbf{d}. \tag{32}$$

The minimum energy solution can be calculated from this quadratic form by solving

$$\tilde{\mathbf{A}}\tilde{\mathbf{y}} = \tilde{\mathbf{b}}. \tag{33}$$

A number of iterative relaxation techniques could be used to implement the minimization of the composite energy (31). For example, we could set the energy gradient of (31) to 0 at each level independently using the current value of $\mathbf{x}$ to compute $\nabla E_d(\mathbf{x}, \mathbf{d})$, and then recompute the summed representation $\mathbf{x}$. Because of the tight coupling between the levels, however, this approach (Gauss-Seidel or Jacobi) does not work very well. Instead, we have found that using conjugate gradient descent performs satisfactorily. To implement this algorithm, we must compute the current residual

$$\tilde{\mathbf{r}} = \tilde{\mathbf{A}}_s \tilde{\mathbf{y}} + \tilde{\mathbf{I}}^T \mathbf{A}_d(\tilde{\mathbf{I}}\tilde{\mathbf{y}} - \mathbf{d}). \tag{34}$$

This is achieved by first computing $\mathbf{x} = \tilde{\mathbf{I}}\tilde{\mathbf{y}}$, then subtracting the data points $\mathbf{d}$ and weighting the error by $\mathbf{A}_d$. This data-fit residual is then converted into the relative representation by

multiplying by $\mathbf{I}^{l^T}$ (the *adjoint* of the interpolation function), and the smoothness component of the residual ($\mathbf{A}_s^l \mathbf{y}^l$) is then added. A similar approach is used to compute the sparse matrix product $\tilde{\mathbf{A}}\tilde{\mathbf{p}}$, where $\tilde{\mathbf{p}}$ is the current direction vector. While this algorithm does not converge as fast as hierarchical basis conjugate gradient, it does produce a multiresolution description while solving the surface interpolation problem.

To reduce the amount of storage required for the interpolation matrices $\mathbf{I}^l$ and the amount of computation performed when interpolating, we can use a recursive structure for the interpolants. We define

$$\mathbf{z}^l = \mathbf{y}^l + \mathbf{I}_{l-1}^l \mathbf{z}^{l-1} \tag{35}$$

as the absolute representation at level l (with $\mathbf{x} = \mathbf{z}_L$), where $\mathbf{I}_{l-1}^l$ is the interpolation function between levels $l-1$ and l. The structure of our relative representation algorithm now looks very similar to that of the hierarchical basis conjugate gradient, except that the smoothness constraint is computed at each level separately, and each level in the pyramid is fully populated. The relative representation thus has more state than the original fine-level system of equations.

Designing a set of spline energies for each level that decomposes $\mathbf{x}$ into a reasonable multiresolution description while maintaining the faithfulness of the energy function (31) to the original energy (15) is a nontrivial task. The discussion in [Szeliski88] gives conditions for the approximation to hold. It is often easier to take an indirect approach to designing these energies using Fourier analysis, where instead of defining the interpolation problem as a functional minimization problem, we view it as a Bayesian estimation problem and try to match the power spectra of the prior models [Szeliski88].

The relative representation as we have described it so far could also be obtained by filtering the fine-level (composite) representation.[9] What may be more interesting is to equip each level with a non-linear smoothness constraint, e.g., one that favors flat or zero surfaces. Instead of making the cost proportional to the square of the surface magnitude, slope or curvature, we could incur a fixed cost whenever one of these quantities is non-zero. For example, we could add terms of the form

$$E_s(\mathbf{x}) = \sum_{(i,j)} (1 - \delta(x_{i,j})) \tag{36}$$

This would favor surfaces which are zero, and higher order terms could be used to favor piecewise constant or piecewise linear solutions.[10] Combining this idea with a multiresolution relative representation, we would obtain an interpolation algorithm which distributes the terrain features among the various levels in a non-linear fashion.[11] While this is still far removed from a true semantic interpretation of our terrain, it may provide the kind of perceptual grouping which is a necessary precursor of object identification [Lowe85].

[9] This is because the levels are a linear function of the input data.

[10] This idea is inspired by a similar constraint in Leclerc's work on minimal-length encoding [Leclerc89], where the optimal polynomial order for a surface patch is determined by penalizing non-zero coefficients.

[11] By contrast, the current relative representation distributes the data among the levels in a "weak" fashion, as if springs of varying stiffness were attached to the surface.

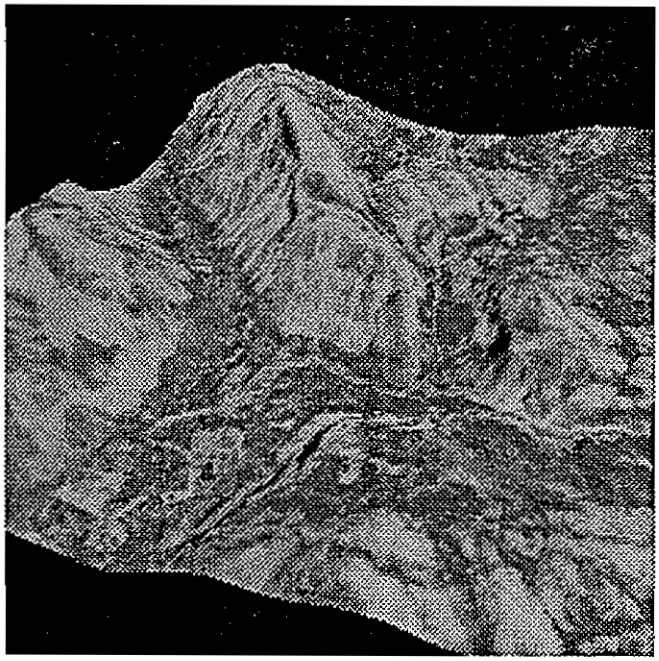


Figure 11: Texture mapped rendering of Yosemite valley

6 Rendering

Once we have interpolated our cartographic data to its final resolution, we may wish to display this information. Cartographic information is often displayed in two-dimensional formats such as contour maps (e.g., Figure 6a). In many cases, however, a three-dimensional perspective view may be more useful for interpreting or analyzing the terrain.

A number of formats can be used for three-dimensional displays. We can use a wireframe with hidden surface removal (Figure 1), but this often limits the available resolution of the display. We can also use a simple shading model to generate a three-dimensional view (Figure 10). Finally, if an intensity image (aerial photograph) is available which corresponds to the terrain model, we can use texture mapping to produce a realistic view of the scene (Figure 11).

The computation of a realistic, properly anti-aliased rendering can be quite complicated and time consuming [Hanson88]. For many applications, however, having a real-time rendering capability greatly enhances the interpretation process. This is especially true of the real-time rotation of three-dimensional surfaces, since a very realistic percept of shape can be obtained.[12] Traditionally, such animations have been available only from graphics workstations[13] or flight simulators. In our own research, we have developed a rendering

[12]In the vision literature, this is called the *kinetic depth effect.*

[13]For example, the top of the line IRIS workstation from Silicon Graphics will render 100,000 polygons per second.

algorithm for the Connection Machine which gives near real-time performance.

6.1 A parallel rendering algorithm

The key to developing a parallel rendering algorithm for the Connection Machine is to assign one processor to each point in the terrain map being rendered. Our algorithm also takes advantage of the combining routing network of the Connection Machine to perform z-buffering. A detailed description of the algorithm follows.

The input to our rendering algorithm consists of two images (2-D arrays): a floating point image of elevation values, and an integer image of intensities to be texture mapped. These latter intensities can be true intensities obtained from aerial photographs, synthetic intensities computed using a shading model, or a simple texture such as a grid. The elevation map is converted into screen coordinates (x, y, and screen-based depth z) using a user-supplied homogeneous transformation matrix [Newman79] (we keep an extra 4 bits of precision on the screen coordinates).

Next, we optionally blur the texture (intensity) image to reduce the aliasing which occurs when multiple points map to the same screen location. To do this, we average each point with its nearest neighbors if these neighbors map to the same or nearby points on the screen (a global parameter called the *blur radius* controls the nearness criterion). We then average the points from the previous step with their neighbors two pixels away if they are still within the selected screen distance. This iteration continues until the aliasing has been eliminated.

At this stage, we can independently draw each pixel into a z-buffer which corresponds to the final image size. To do this, we append the intensity value for each map point to its depth (z) value, i.e., we form a long (32-bit) integer which contains the z value in its high-order bits and the intensity in its low-order bits. Using the global routing network of the Connection Machine with *max* combining[14] we can directly draw each pixel into the z-buffer (pixels falling outside the z-buffer are automatically clipped). The low-order bits of the z-buffer then correspond to the rendered image.

This "simple" rendering is fast, but it often leaves gaps in the image where a map grid square covers more than one pixel. While this fast rendering may be suitable for quick interactive positioning and for obtaining a depth percept, a better algorithm must be devised for final renderings. We designed our "complex" rendering algorithm by assigning a processor to each square interior to the digital map (this is basically equivalent to the node per processor assignment with some replication at the borders). Each processor then checks if all four corners of the square map to the same screen location. If so, the pixel is drawn; if not, the square is subdivided (refined) before drawing.

The subdivision occurs by first determining the maximum number of screen pixels separating the square corners along each dimension

$$m_{i,j} = \max(|x_{i+1,j} - x_{i,j}| + |y_{i+1,j} - y_{i,j}|, |x_{i+1,j+1} - x_{i,j+1}| + |y_{i+1,j+1} - y_{i,j+1}|, 1)$$

[14]The *Lisp instruction is `(*pset :max ...)`.

$$n_{i,j} = \max(|x_{i,j+1} - x_{i,j}| + |y_{i,j+1} - y_{i,j}|, |x_{i+1,j+1} - x_{i+1,j}| + |y_{i+1,j+1} - y_{i+1,j}|, 1).$$

We then subdivide each square into $m_{i,j} \times n_{i,j}$ pieces, thus guaranteeing that each piece covers at most one pixel. To implement the subdivision, each square processor "grabs" $s_{i,j} = m_{i,j}n_{i,j}$ processors using a (scan!! $s_{i,j}$ '+!!) global scan to determine the starting address of the group of processors it will use. Often, there are not enough processors available in the rendering processor set, in which case the complete subdivision/rendering stage may take several iterations. Each group of processors renders its assigned square in parallel by first computing the screen addresses and intensity using bilinear interpolation and then calling the "simple" rendering procedure.[15]

The performance which we have achieved with this algorithm is quite impressive. On our 8K Connection Machine running unoptimized *Lisp code, we can render a 256×256 texture-mapped image in 0.45 seconds for the simple version and 2.61 seconds for the complex fully interpolated version (these figures vary, depending on the particular viewing position).

There are a number of extensions possible to our rendering algorithm which we have not implemented. Instead of mapping an intensity image onto the terrain map, we could map the surface normals. This would allow us to move the light source position and recompute the synthetic shaded image in real time, which would aid in the perception of surface shape.[16] We could also generalize the algorithm to render arbitrary 3-D surfaces defined using polygonal (quadrilateral) surface patches. This would only require a different front end to the algorithm, since the same routines could be used for coordinate transformation, quadrilateral subdivision, and z-buffer rendering. Finally, if we are to achieve the same quality of renderings as is currently available with the SRI Cartographic Modeling Environment [Hanson88], we would have to add a multiresolution pyramid representation of the terrain and intensity to our algorithm. Integrating this representation into our current algorithm is difficult, since it is based on projecting each map point into the rendered image rather than ray tracing screen points back to the map. However, a recursive rendering algorithm based on the idea of quadtrees [Rosenfeld80] may be feasible.

7 Discussion

The approach to surface interpolation developed in this paper is based on using a regular grid representation for the terrain map. This representation fits in well with the usual desired output of the interpolation stage. It is limited, however, to representing single-valued functions such as terrain (for a description of how to integrate this representation with three-dimensional models of buildings and objects, see [Hanson88, Bobick89].).

[15]Another alternative which we considered was iteratively splitting each square into two until it is small enough. This approach, however, requires keeping a list of "active" squares which have not yet been painted, and ensuring that this list does not overflow existing memory is difficult.

[16]This suggestion was made by Yvan Leclerc.

The spline interpolator defined over this regular grid results in a very large sparse set of equations which must be solved iteratively. Fortunately, multiresolution acceleration techniques (in particular, our hierarchical basis conjugate gradient algorithm) make the solution of these equations feasible in a reasonable amount of time.

These iterative relaxation algorithms are inherently parallel, and are therefore well matched to massively parallel architectures such as the Connection Machine. Multiresolution algorithms, however, can sometimes underutilize the parallelism in such an architecture, especially if the coarser levels of the pyramid are smaller than the processor array size.[17] Because of their regularity, our parallel algorithms should also execute efficiently on sequential, vector, or MIMD (coarse-grained) parallel architectures. The key to the widespread availability and acceptance of our parallel relaxation algorithms will probably lie in our ability to automatically covert code from a high-level parallel language such as *Lisp into languages such as Common Lisp, C, or FORTRAN than can be compiled on more conventional architectures. This promises to be an interesting area for future research.

8 Conclusions

In this paper, we have demonstrated how variational splines can be used for performing surface interpolation, which is an important problem in digital cartography. Our variational splines are defined using an optimization framework, where the characteristics of the desired solution are specified rather than the algorithm used to compute the solution. This optimization framework is useful for integrating data from multiple sources and resolutions, and also allows us to locally control surface characteristics such as continuity and smoothness.

To compute the variational spline, we use a fine, regular discretization which corresponds in resolution to the final digital map we wish to produce. This discretization results in a simple local set of equations and is well suited to parallel representations and algorithms. To efficiently solve these equations, we must use some multiresolution acceleration technique. In our experience, conjugate gradient descent preconditioned with hierarchical basis functions has proven to be the most efficient method.

We have applied our surface interpolation algorithm to the problem of data amplification, i.e., obtaining a dense cartographic representation from a sparse set of samples. Examples with both sparse (punctate) data and contour lines were presented. We also showed how to locally control the surface, for example by introducing creases along hydrographic features. Natural looking fractal detail can be added to our interpolated solution through stochastic relaxation. The random samples thus generated can be used to quantify the local uncertainty (variance) in our terrain model estimates.

The problem of data reduction was also studied briefly. We described a relative representation surface interpolation algorithm which decomposes the surface into a hierarchy of

[17] In this case, the combination of a smaller parallel machine and *virtual processors* [TMC88] may actually be a boon.

scales, and suggested how this could be extended to perform a more non-linear feature extraction. In the final part of the paper, we developed a parallel three-dimensional rendering algorithm which can be used to visualize our interpolated surfaces.

As we have demonstrated in this paper, manipulating terrain maps using parallel relaxation algorithms gives the user more flexibility in integrating data and adding local constraints. The advent of parallel architectures and the development of fast parallel algorithms is quickly making this approach practical for real cartographics applications.

References

[Ahlberg67] J. H Ahlberg, E. N. Nilson, and J. L. Walsh. *The Theory of Splines and their Applications.* Academic Press, New York, New York, 1967.

[Anderssen74] R. S. Anderssen and P. Bloomfield. A time series approach to numerical differentiation. *Technometrics*, 16(1):69–75, February 1974.

[Bartels87] R. H. Bartels, J. C. Beatty, and B. A. Barsky. *An Introduction to Splines for use in Computer Graphics and Geeometric Modeling.* Morgan Kaufmann Publishers, Los Altos, California, 1987.

[Blake87] A. Blake and A. Zisserman. *Visual Reconstruction.* MIT Press, Cambridge, Massachusetts, 1987.

[Bobick89] A. F. Bobick and R. C. Bolles. Representation space: An approach to the integration of visual information. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'89)*, pages 492–493, San Diego, California, June 1989. IEEE Computer Society Press.

[Boult86] T. E. Boult. *Information Based Complexity in Non-Linear Equations and Computer Vision.* PhD thesis, Columbia University, 1986.

[Briggs87] W. L. Briggs. *A Multigrid Tutorial.* Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1987.

[Burt83] P. J. Burt and E. H. Adelson. The Laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, COM-31(4):532–540, April 1983.

[Cendes87] Z. J. Cendes and S. H. Wong. C^1 quadratic interpolation over arbitrary point sets. *IEEE Computer Graphics and Applications*, 7(11):8–16, November 1987.

[Craven79] P. Craven and G. Wahba. Smoothing noisy data with spline functions: Estimating the correct degree of smoothing by the method of generalized cross-validation. *Numerische Mathematik*, 31:377–403, 1979.

[Crowley82] J. L. Crowley and R. M. Stern. Fast computation of the difference of low-pass transform. Technical Report CMU-RI-TR-82-18, The Robotics Institute, Carnegie Mellon University, November 1982.

[Fischler89] M. A. Fischler and T. M. Strat. Recognizing objects in a natural environment: A contextual vision system (CVS). In *Image Understanding Workshop*, pages 774–796, Palo Alto, California, May 1989. Morgan Kaufmann Publishers.

[Foley87] T. A. Foley. Weighted bicubic spline interpolation to rapidly varying data. *ACM Transactions on Graphics*, 6(1):1–18, January 1987.

[Fournier82] A. Fournier, D. Fussel, and L. Carpenter. Computer rendering of stochastic models. *Communications of the ACM*, 25(6):371–384, 1982.

[Fuchs77] H. Fuchs, Z. M. Kedem, and S. P. Uselton. Optimal surface reconstruction from planar contours. *Communications of the ACM*, 20(10):693–702, October 1977.

[Geman84] S. Geman and D. Geman. Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, November 1984.

[Hackbusch85] W. Hackbusch. *Multigrid Methods and Applications.* Springer-Verlag, Berlin, 1985.

[Hannah88] M. J. Hannah. Test results from SRI's stereo system. In *Image Understanding Workshop*, pages 740–744, Cambridge, Massachusetts, April 1988. Morgan Kaufmann Publishers.

[Hanson88] A. J. Hanson and L. H. Quam. Overview of the SRI cartographic modeling environment. In *Image Understanding Workshop*, pages 576–582, Cambridge, Massachusetts, April 1988. Morgan Kaufmann Publishers.

[Hewett86] T. A. Hewett. Fractal distributions of reservoir heterogeneity and their influence on fluid transport. In *SPE 15386, 61st Annual Technical Conference and Exibition of the Society of Petroleum Engineers*, New Orleans, Louisiana, October 1986. Society of Petroleum Engineers.

[Hillis85] W. D. Hillis. *The Connection Machine.* MIT Press, Cambridge, Massachusetts, 1985.

[Kimeldorf70] G. Kimeldorf and G. Wahba. A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. *The Annals of Mathematical Statistics*, 41(2):495–502, 1970.

[Leclerc89] Y. G. Leclerc. Constructing simple stable descriptions for image partitioning. *International Journal of Computer Vision*, 3(1):75–104, 1989.

[Lowe85] D. G. Lowe. *Perceptual Organization and Visual Recognition.* Kluwer Academic Publishers, Boston, Massachusetts, 1985.

[Mandelbrot82] B. B. Mandelbrot. *The Fractal Geometry of Nature.* W. H. Freeman, San Francisco, California, 1982.

[Marroquin84] J. L. Marroquin. Surface reconstruction preserving discontinuities. A. I. Memo 792, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, August 1984.

[Marroquin85] J. L. Marroquin. *Probabilistic Solution of Inverse Problems.* PhD thesis, Massachusetts Institute of Technology, 1985.

[Maybeck82] P. S. Maybeck. *Stochastic Models, Estimation, and Control*, volume 2. Academic Press, New York, New York, 1982.

[Newman79] W. M. Newman and R. F. Sproull. *Principles of Interactive Computer Graphics.* McGraw Hill, New York, New York, 2 edition, 1979.

[Oppenheim75] A. V. Oppenheim and R. W. Schafer. *Digital Signal Processing.* Prentice Hall, Englewood Cliffs, New Jersey, 1975.

[Pentland84] A. P. Pentland. Fractal-based description of natural scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):661–674, November 1984.

[Platt88] J. C. Platt and Alan H. Barr. Constrained differential optimization. In *Neural Information Processing Systems, Denver, Colorado, 1987*, pages 612–621, New York, New York, 1988. American Institute of Physics.

[Poggio85a] T. Poggio, V. Torre, and C. Koch. Computational vision and regularization theory. *Nature*, 317(6035):314–319, 26 September 1985.

[Poggio85b] T. Poggio, H. Voorhees, and A. Yuille. A regularized solution to edge detection. A. I. Memo 833, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, May 1985.

[Potter85] J. L. Potter, editor. *The Massively Parallel Processor.* MIT Press, Cambridge, Massachusetts, 1985.

[Press86] W. Press et al. *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press, Cambridge, England, 1986.

[Rosenfeld80] A. Rosenfeld. Quadtrees and pyramids for pattern recognition and image processing. In *Fifth International Conference on Pattern Recognition (ICPR'80)*, pages 802–809, Miami Beach, Florida, December 1980. IEEE Computer Society Press.

[Slama80] Chester C. Slama, editor. *Manual of Photogrammetry.* American Society of Photogrammetry, Falls Church, Virginia, fourth edition, 1980.

[Szeliski87] R. Szeliski. Regularization uses fractal priors. In *Sixth National Conference on Artificial Intelligence (AAAI-87)*, pages 749–754, Seattle, Washington, July 1987. Morgan Kaufmann Publishers.

[Szeliski88] R. Szeliski. *Bayesian Modeling of Uncertainty in Low-Level Vision.* PhD thesis, Carnegie Mellon University, August 1988.

[Szeliski89a] R. Szeliski. Fast surface interpolation using hierarchical basis functions. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'89)*, pages 222–228, San Diego, California, June 1989. IEEE Computer Society Press.

[Szeliski89b] R. Szeliski and D. Terzopoulos. From splines to fractals. *Computer Graphics (SIGGRAPH'87)*, 23(4), July 1989.

[Terzopoulos83] D. Terzopoulos. Multilevel computational processes for visual surface reconstruction. *Computer Vision, Graphics, and Image Processing*, 24:52–96, 1983.

[Terzopoulos86a] D. Terzopoulos. Image analysis using multigrid relaxation methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(2):129–139, March 1986.

[Terzopoulos86b] D. Terzopoulos. Regularization of inverse visual problems involving discontinuities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(4):413–424, July 1986.

[Tikhonov77] A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill-Posed Problems.* V. H. Winston, Washington, D. C., 1977.

[TMC88] Supplement to the *Lisp reference manual, version 5.0. Technical report, Thinking Machines Corporation, Cambridge, Massachusetts, September 1988.

[USGS86] Digital line graphs from 1:24,000-scale maps. National Mapping Program Technical Instructions, Data Users Guide 1, United States Department of the Interior, U. S. Geological Survey, Reston, Virginia, 1986.

[USGS87] Digital elevation models. National Mapping Program Technical Instructions, Data Users Guide 5, United States Department of the Interior, U. S. Geological Survey, Reston, Virginia, 1987.

[Voss85] R. F. Voss. Random fractal forgeries. In R. A. Earnshaw, editor, *Fundamental Algorithms for Computer Graphics.* Springer-Verlag, Berlin, 1985.

[Young71] D. Young. *Iterative Solution of Large Linear Systems.* Academic Press, New York, New York, 1971.

[Yserentant86] H. Yserentant. On the multi-level splitting of finite element spaces. *Numerische Mathematik*, 49:379–412, 1986.

A Finite element equations

In this Appendix, we present the discrete implementation of the energy equations used in Section 3.1. The energy equations which we develop are formulated in terms of a number of discrete fields which form the basic data structures used in the surface interpolation algorithm:

$x_{i,j}$	surface depth (free variables)
$d_{i,j}$	depth constraints
$c_{i,j}$	depth constraint weights (σ^{-2})
$l_{i,j}, m_{i,j}$	depth discontinuities [0,1]
$n_{i,j}$	orientation discontinuities [0,1]

The line variables $l_{i,j}$ and $m_{i,j}$ are located on a dual grid, and the crease variables $n_{i,j}$ are coincident with the depth value nodes.

To describe the discrete version of the smoothness constraint, we define a number of *implicit* fields which are not actually computed by the algorithm but serve only as a notational shorthand. First, we define the finite differences

$$\begin{aligned}
x^{u}_{i,j} &= x_{i+1,j} - x_{i,j} \\
x^{v}_{i,j} &= x_{i,j+1} - x_{i,j} \\
x^{uu}_{i,j} &= x^{u}_{i+1,j} - x^{u}_{i,j} = x_{i+2,j} - 2x_{i+1,j} + x_{i,j} \\
x^{uv}_{i,j} &= x^{u}_{i,j+1} - x^{u}_{i,j} = x^{v}_{i+1,j} - x^{v}_{i,j} = x_{i+1,j+i} - x_{i+1,j} - x_{i,j+1} + x_{i,j} \\
x^{vv}_{i,j} &= x^{v}_{i,j+1} - x^{v}_{i,j} = x_{i,j+2} - 2x_{i,j+1} + x_{i,j}.
\end{aligned}$$

Next, we define the continuity strengths

$$\begin{aligned}
\beta^{u}_{i,j} &= (1 - l_{i,j}) \\
\beta^{v}_{i,j} &= (1 - m_{i,j}) \\
\beta^{uu}_{i,j} &= (1 - l_{i,j})(1 - l_{i+1,j})(1 - n_{i+1,j}) \\
\beta^{uv}_{i,j} &= (1 - l_{i,j})(1 - l_{i,j+1})(1 - m_{i,j})(1 - m_{i+1,j})(1 - n_{i,j}n_{i+1,j+1})(1 - n_{i+1,j}n_{i,j+1}) \\
\beta^{vv}_{i,j} &= (1 - m_{i,j})(1 - m_{i,j+1})(1 - n_{i,j+1}).
\end{aligned}$$

These continuity strengths determine which of the *molecules* defined by the $x^{u}_{i,j}, \ldots, x^{vv}_{i,j}$ finite differences are active (we use phantom line variables around the edges of the grid to take care of boundary conditions).

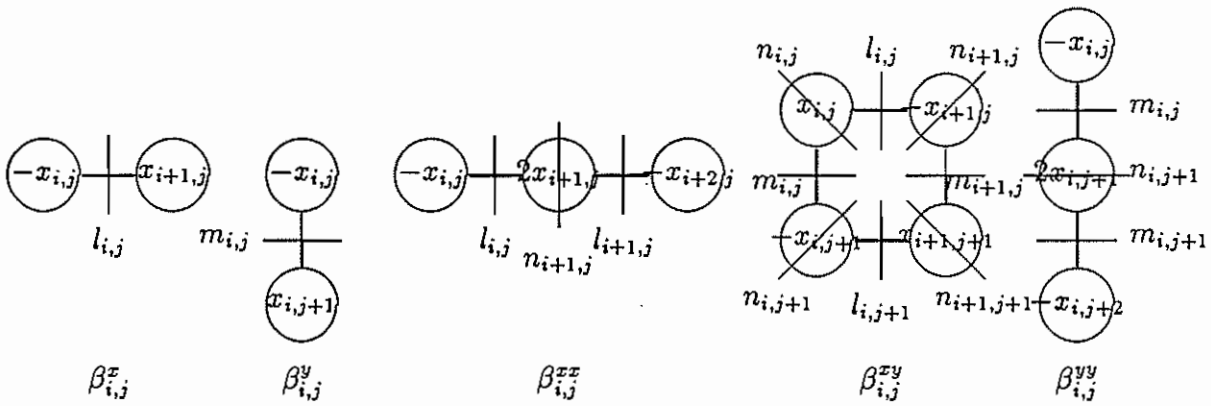


Figure 12: Continuity strengths and computational molecules

The interactions between the depth field, the line and crease variables, and the continuity strengths are shown in Figure 12. Intuitively, the line variables disable any molecules that are cut when the line variable is turned on. The crease variables disable the $x^{uu}_{i,j}$ or $x^{vv}_{i,j}$ molecules whose centers are coincident with the crease, and disable the $x^{uv}_{i,j}$ molecules if two opposite corners are creased.

The prior energy (weak smoothness constraint) is constructed from these elements

$$E_s(\mathbf{x}) = \frac{1}{2}\sum_{(i,j)} \begin{array}{l} w_0 h^2 \left[(x_{i,j})^2\right] + w_1 \left[\beta^u_{i,j}(x^u_{i,j})^2 + \beta^v_{i,j}(x^v_{i,j})^2\right] \\ + w_2 h^{-2}\left[\beta^{uu}_{i,j}(x^{uu}_{i,j})^2 + 2\beta^{uv}_{i,j}(x^{uv}_{i,j})^2 + \beta^{vv}_{i,j}(x^{vv}_{i,j})^2\right] \end{array} \tag{37}$$

where $h = \Delta u = \Delta v$ is the grid spacing. The weighting functions w_m, which come from the general controlled-continuity constraint (4), define the order of the interpolator. In terms of the notation used by Terzopoulos [Terzopoulos86b] which is shown in (3), we have

$$\{w_0, w_1, w_2\} = \{0, \rho(u,v)[1-\tau(u,v)], \rho(u,v)\tau(u,v)\}$$

(in our implementation, we use the line and crease variables to represent the discontinuities rather than spatially varying w_m).

The equation for the data compatibility constraint is the same as we presented in Section 3.1, namely

$$E_d(\mathbf{x}, \mathbf{d}) = \frac{1}{2}\sum_{(i,j)} c_{i,j}(x_{i,j} - d_{i,j})^2 \tag{38}$$

with $c_{i,j} = \sigma^{-2}_{i,j}$ being the inverse variance of the measurement $d_{i,j}$.

The stiffness matrix $\mathbf{A}$ and the weighted data vector $\mathbf{b}$ can be obtained from the discrete energy equation

$$E(\mathbf{x}) = E_s(\mathbf{x}) + E_d(\mathbf{x}, \mathbf{d})$$

by setting

$$a_{(i,j),(k,l)} = \left. \frac{\partial^2 E(\mathbf{x})}{\partial x_{i,j} \partial x_{k,l}} \right|_{\mathbf{x}=\mathbf{0}} \tag{39}$$

and

$$b_{i,j} = - \left. \frac{\partial E(\mathbf{x})}{\partial x_{i,j}} \right|_{\mathbf{x}=\mathbf{0}} . \tag{40}$$

The **A** matrix and **b** vector are precomputed before the relaxation begins. Since **A** has at most 13 non-zero entries per row, we can store these coefficients in an $M \times N \times 13$ array, where M and N are the terrain model dimensions.

Acknowledgements

This research was supported by a Visiting Scientist position at the Artificial Intelligence Center of SRI International. I would like to thank Martin Fischler, Yvan Leclerc, Lynn Quam and the other members of the Perception Group for their ideas and encouragement.